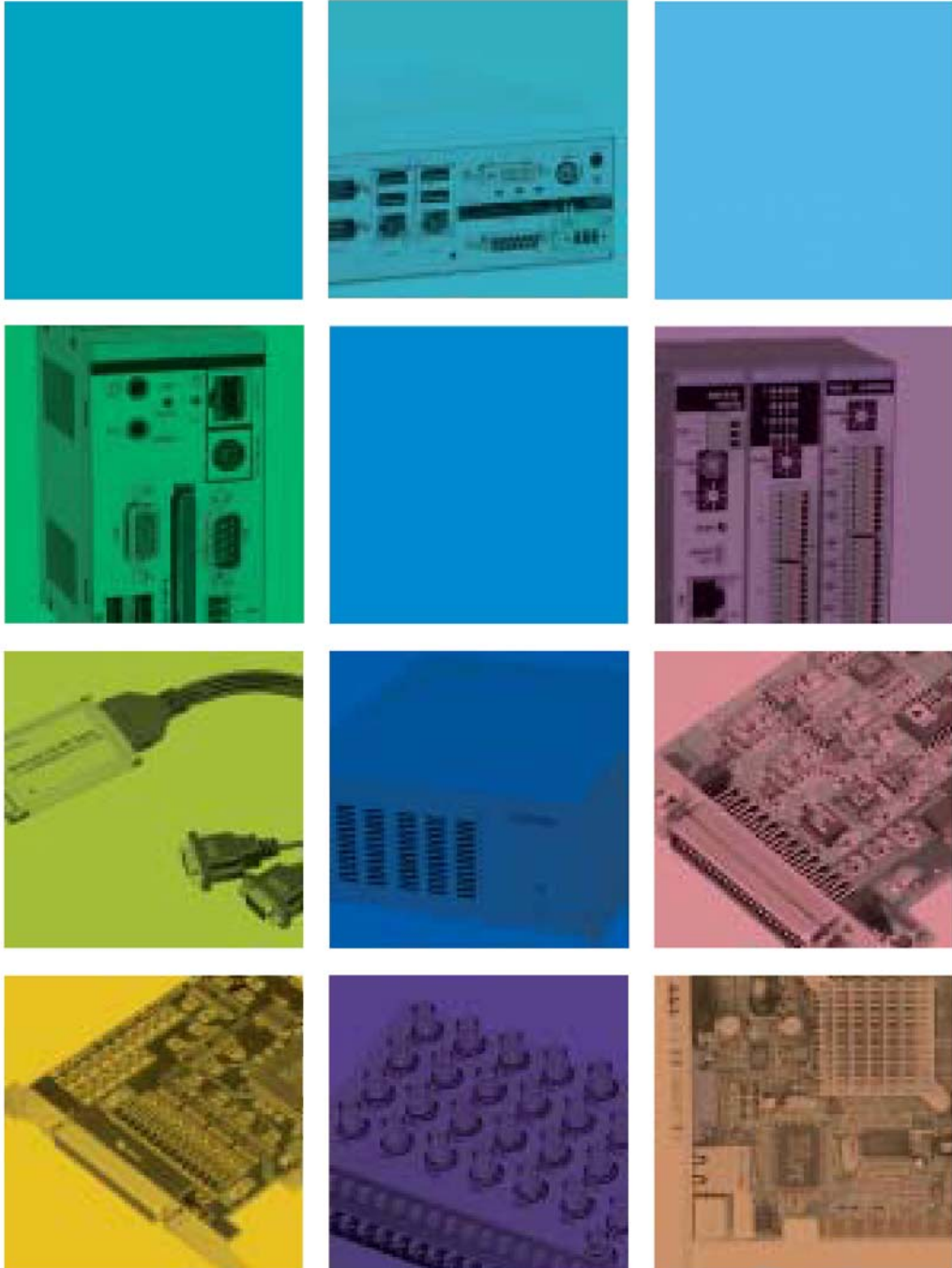


アナログ入出力 ビギナーズ・ガイドブック

Visual Basic2005プログラミング編



株式会社コンテック

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『ヒギナーズ・ガイドブック』を発行する事となりました。本書では、アナログ入出力の概要と用語解説、弊社製アナログ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4／5章 Visual Basic6.0によるアナログ入出力プログラミング／ActiveXによるコンポーネントプログラミング

- ① OS／開発言語 : Microsoft Windows XP Professional・Home Edition／Microsoft Visual Basic 2005
- ② インターフェイス : CardBus対応非絶縁型低価格高精度アナログ入出力カード 【型式: ADA16-8/2 (CB) L】
- ③ ケーブル : 68ピン→50ピン変換シールドケーブル 【型式: ADC-68M/50M】
- ④ アクセサリ : BNC端子台 【型式: ATP-8L】
: 圧着用中継端子台（参考資料でのみ使用） 【型式: EPD-50A】
- ⑤ ソフトウェア : Windows版高機能アナログ入出力ドライバソフトウェア（製品添付） 【型式: API-AIO (WDM)】
: 計測システム開発用ActiveXコンポーネント集（別売：体験版有り） 【型式: ACX-PAC (W32) Ver.4.1】

構成例 ①



構成例 ②

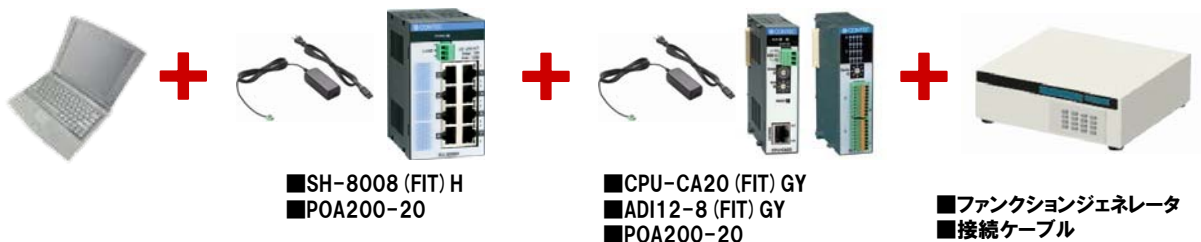
ハードウェアが用意できない場合は、ドライバソフトウェアのデモボード機能で実習体験ができます。



● 第6章 リモートI/O

- ① OS／開発言語 : Microsoft Windows XP Professional・Home Edition／Microsoft Visual Basic 6.0
- ② 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（絶縁型アナログ入力モジュール） 【型式: ADI12-8 (FIT) GY】
: ACアダプタ電源 ×2台 【型式: POA200-20】
: スイッチングHUBユニット 【型式: SH-8008 (FIT) H】
- ③ 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4章 Visual Basic2005によるアナログ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:4-12頁]
- ② タイマコントロールによる簡易連続アナログ入力プログラム [プログラムリスト:4-25頁]
- ③ タイマコントロールによる簡易連続アナログ入力プログラム (ファイル保存機能付) [プログラムリスト:4-30頁]
- ④ FIFOメモリを使用した高速サンプリング (簡易オシロスコープ:データ表示) [プログラムリスト:4-48頁]
- ⑤ FIFOメモリを使用した高速サンプリング (簡易オシロスコープ:波形表示) [プログラムリスト:4-65頁]
- ⑥ タイマコントロールによる簡易連続アナログ出力プログラム [プログラムリスト:4-78頁]
- ⑦ FIFOメモリを使用した高速連続アナログ出力プログラム [プログラムリスト:4-95頁]

● 第5章 ActiveXによるコンポーネントプログラミング

- ① FIFOメモリを使用した『簡易オシロスコープ』 [プログラムリスト:5-12頁]

● 第6章 リモートI/O

- ① リモートI/Oによるアナログ入力プログラミング [プログラムリスト:6-27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、コンテックホームページにて → <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。また、本書にて記載している一部のプログラムは、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。
- ※ アナログ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/>
- ※ 計測システム開発用ActiveXコンポーネント集『ACX-PAC (W32) Ver.4.1』の体験版は弊社ホームページからご請求できます。
<http://www.contec.co.jp/acxpac/>

目次

第1章 パソコンによる計測・制御システム	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
第2章 アナログ入出力の基礎知識	2-2
2-1. アナログ入出力とは	2-2
2-2. アナログ入出力ボード／カードの分類	2-2
2-3. アナログ信号のデジタル化、デジタル信号のアナログ化	2-3
2-4. アナログ入出力ボード／カードの絶縁タイプ	2-4
2-4-1. バス絶縁型	2-4
2-4-2. 独立絶縁(チャンネル間絶縁)型	2-4
2-5. 用語解説	2-5
2-5-1. 入出力チャンネル数	2-5
①シングルエンド入力	2-5
②差動入力	2-5
2-5-2. 分解能	2-6
2-5-3. 入出力レンジ	2-6
2-5-4. ゲイン	2-6
2-5-5. 変換速度(サンプリング周期)	2-7
TOPICS：サンプリングの定理	2-7
2-5-6. 変換精度	2-7
2-5-7. バイナリデータと電圧値の関係(分解能16ビットの場合)	2-8
TOPICS：略語の意味(LSB、MSB、FSR)	2-8
①ストレート・バイナリ	2-9
②オフセット・バイナリ	2-9
③コンプリメント・バイナリ(2の補数)	2-9
2-5-8. サンプリング方式	2-10

TOPICS : 変換速度とチャネル数の関係	2-10
TOPICS : 同時出力方式のイメージ(弊社アナログ出力ボード/カード)	2-10
2-5-9. クロック	2-11
①内部クロック	2-11
②外部クロック	2-11
③ソフトウェアクロック	2-11
2-5-10. トリガ	2-12
①ソフトウェアトリガ	2-12
②外部トリガ	2-12
③レベル比較(変換データ比較)トリガ	2-12
2-5-11. バッファメモリ	2-13
①FIFO形式	2-13
②RING(リング)形式	2-13
2-5-12. バスマスタ転送機能	2-14
2-5-13. I/OポートとI/Oポートアドレス	2-15
2-5-14. 割り込み	2-15
2-5-15. 消費電流	2-16
2-5-16. 参考資料 : ノイズの種類とその対策	2-16
①外来ノイズ	2-16
②内部ノイズ	2-16
対策	2-16

第3章 ハードウェア・ソフトウェアのセットアップ 3-2

3-1. 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)	3-2
3-2. 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	3-3
3-3. 手順② : ハードウェアの設定	3-4
3-4. 手順③ : ハードウェアのインストール	3-5
3-5. 手順④ : ドライバソフトウェアの初期設定	3-6
3-6. 手順⑤ : 外部機器との接続	3-7
3-6-1. 外部機器との接続形態(接続方法)	3-7
3-6-2. 本書にて使用する外部機器	3-7
3-6-3. 実習環境の構築	3-8
3-6-4. 参考資料 : 圧着用端子台を使用して外部機器と接続する方法	3-9
3-7. 手順⑥ : ハードウェア・ソフトウェアの動作確認	3-10
3-7-1. 確認方法	3-10
3-8. DEMO DEVICE(仮想ハードウェア)の登録・設定方法	3-11
3-8-1. デモボードの登録方法	3-11
3-8-2. 補足	3-12
3-9. 参考資料(ドライバソフトウェアの必要性)	3-13
3-10. 参考資料(DLL : Dynamic Link Libraryとは)	3-14
3-10-1. Windows環境における実行ファイル(.EXEファイル)	3-14

第4章 Visual Basic2005によるアナログ入出力プログラミング	4-2
4-1. Visual Basicでのプログラム開発	4-2
4-1-1. イベント駆動型(イベントドリブン)言語とは	4-2
4-1-2. Visual Basic6.0用語解説	4-3
4-1-3. Visual Basicのプログラム作成手順	4-3
4-2. 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	4-4
4-2-1. プログラムフローチャート	4-4
4-2-2. プログラム作成手順①(Visual Basicの起動)	4-4
4-2-3. プログラム作成手順②(コントロールの配置)	4-6
4-2-4. プログラム作成手順③(フォームのプロパティ設定)	4-7
4-2-5. 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)	4-7
4-2-6. 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)	4-8
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	4-8
4-2-7. 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)	4-9
TOPICS : タイマコントロールに関して	4-9
4-2-8. プログラム作成手順⑦(タイマ開始処理の記述)	4-10
TOPICS : オブジェクトの指定方法とプロシージャの指定方法	4-10
TOPICS : Visual Basicのインテリセンス機能	4-10
4-2-9. プログラム作成手順⑧(タイマ停止処理の記述)	4-11
4-2-10. 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)	4-11
4-2-11. カウントアッププログラムリスト	4-12
4-2-12. プログラム作成手順⑩(プログラムの保存と動作の確認)	4-13
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	4-13
4-3. タイマコントロールによる簡易連続アナログ入力プログラム	4-14
4-3-1. プログラム概要	4-14
4-3-2. 実行環境	4-14
4-3-3. プログラムフローチャート	4-15
TOPICS : API-AIO(WDM)について	4-15
4-3-4. 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)	4-16
4-3-5. プログラム作成手順②(標準モジュールファイルの追加)	4-16
4-3-6. プログラム作成手順③(変数の追加)	4-17
4-3-7. プログラム作成手順④(初期化処理とアナログ入力初期化処理の追加)	4-17
TOPICS : デバイスハンドルとは	4-18
4-3-8. プログラム作成手順⑤(終了処理の追加)	4-20
4-3-9. プログラム作成手順⑥(タイマ開始処理の記述)	4-21
4-3-10. プログラム作成手順⑦(タイマ停止処理の記述)	4-21
4-3-11. プログラム作成手順⑧(アナログ入力処理の追加)	4-22
4-3-12. プログラムの実行	4-23
4-3-13. タイマコントロールによる簡易連続アナログ入力プログラムリスト	4-25
4-3-14. プログラムの改造(データロギング(ファイル保存)機能の付加)	4-26
TOPICS : Microsoft CommonDialog Controlについて	4-27
TOPICS : Visual Basic6.0でのファイルのオープン/書き込み/クローズの方法	4-29

4-4. FIFOメモリを使用した高速サンプリング(簡易オシロスコープ：データ表示)	4-32
4-4-1. プログラム概要	4-32
4-4-2. 実行環境	4-32
4-4-3. プログラムフローチャート	4-33
4-4-4. 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)	4-34
4-4-5. プログラム作成手順②(標準モジュールファイルの追加)	4-34
4-4-6. プログラム作成手順③(割り込み処理の実現：Cmessageコントロールの追加)	4-35
4-4-7. プログラム作成手順④(変数の追加)	4-38
4-4-8. プログラム作成手順⑤(初期化処理とアナログ入力デバイスリセットの追加)	4-38
4-4-9. プログラム作成手順⑥(終了処理の追加)	4-40
4-4-10. プログラム作成手順⑦(変換条件設定処理の追加)	4-41
4-4-11. プログラム作成手順⑧(メモリ領域のリセットと変換開始命令の追加)	4-46
4-4-12. プログラム作成手順⑨(イベント発生時処理(データ取得処理)の追加)	4-47
4-4-13. プログラムの実行	4-48
4-4-14. FIFOメモリを使用した高速サンプリング(簡易オシロスコープ：データ表示) リスト	4-48
4-5. FIFOメモリを使用した高速サンプリング(簡易オシロスコープ：グラフ表示)	4-50
4-5-1. プログラム概要	4-50
4-5-2. 実行環境	4-50
4-5-3. プログラムフローチャート	4-51
4-5-4. 参考資料：Visual Basic6.0のピクチャーボックスコントロールの基礎知識	4-52
4-5-5. 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)	4-52
4-5-6. プログラム作成手順②(標準モジュールファイルの追加)	4-52
4-5-7. プログラム作成手順③(割り込み処理の実現：Cmessageコントロールの追加)	4-53
4-5-8. プログラム作成手順④(変数の追加)	4-54
4-5-9. プログラム作成手順⑤(初期化処理とアナログ入力デバイスリセットの追加)	4-54
4-5-10. プログラム作成手順⑥(終了処理の追加)	4-56
4-5-11. プログラム作成手順⑦(変換条件設定処理の追加)	4-57
4-5-12. プログラム作成手順⑧(メモリ領域のリセットと変換開始、グラフ初期化処理の追加)	4-62
4-5-13. プログラム作成手順⑨(イベント発生時処理(データ取得処理)の追加)	4-63
4-5-14. プログラムの実行	4-64
TOPICS：もっと、手軽に簡単にプログラムを組みたい方へ	4-64
4-5-15. FIFOメモリを使用した高速サンプリング(簡易オシロスコープ：グラフ表示) リスト	4-65
4-6. タイマコントロールによる簡易連続アナログ出力プログラム	4-67
4-6-1. プログラム概要	4-67
4-6-2. 実行環境	4-67
4-6-3. プログラムフローチャート	4-68
TOPICS：API-AIO(WDM)について	4-68
4-6-4. 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)	4-69
4-6-5. プログラム作成手順②(標準モジュールファイルの追加)	4-69
4-6-6. プログラム作成手順③(変数の追加)	4-70
4-6-7. プログラム作成手順④(初期化処理とアナログ出力初期設定、変数初期化の追加)	4-70
TOPICS：デバイスハンドルとは	4-71
4-6-8. プログラム作成手順⑤(終了処理の追加)	4-73
4-6-9. プログラム作成手順⑥(タイマ開始処理の記述)	4-74
4-6-10. プログラム作成手順⑦(タイマ停止処理の記述)	4-74

4-6-11. プログラム作成手順⑧(アナログ出力処理の追加)	4-75
4-6-12. プログラムの実行	4-76
4-6-13. タイマコントロールによる簡易連続アナログ出力プログラムリスト	4-78
4-7. FIFOメモリを使用した高速連続アナログ出力プログラム	4-79
4-7-1. プログラム概要	4-79
4-7-2. 実行環境	4-79
4-7-3. プログラムフローチャート	4-80
4-7-4. 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)	4-81
4-7-5. プログラム作成手順②(標準モジュールファイルの追加)	4-81
4-7-6. プログラム作成手順③(割り込み処理の実現: Cmessageコントロールの追加)	4-82
4-7-7. プログラム作成手順④(変数の追加)	4-83
4-7-8. プログラム作成手順⑤(初期化処理とアナログ出力デバイスリセットの追加)	4-86
4-7-9. プログラム作成手順⑥(終了処理の追加)	4-85
4-7-10. プログラム作成手順⑦(変換条件設定処理の追加)	4-86
4-7-11. プログラム作成手順⑧(アナログ出力開始処理の追加)	4-92
4-7-12. プログラム作成手順⑨(イベント発生時処理の追加)	4-93
4-7-13. プログラムの実行	4-94
4-7-14. FIFOメモリを使用した高速連続アナログ出力プログラムリスト	4-95
4-8. ドライバソフトウェア提供関数一覧	4-97

第5章 ActiveXによるコンポーネントプログラミング 5-2

5-1. コンポーネントプログラミングとは	5-2
5-2. ActiveXコントロールとは	5-2
5-3. 計測システム開発用ActiveXコンポーネント集『ACX-PAC(W32)』紹介	5-3
5-4. アナログ入出力カードのFIFOメモリを使用した『簡易オシロスコープ』の作成	5-4
5-4-1. プログラム概要	5-4
5-4-2. 実行環境	5-4
5-4-3. プログラム作成手順①(Visual Basicの起動)	5-5
5-4-4. プログラム作成手順②(ActiveXコントロールの追加と各コントロールの貼り付け)	5-6
5-4-5. プログラム作成手順③(各コントロールのプロパティ設定)	5-7
5-4-6. プログラム作成手順④(コードの記述)	5-10
5-4-7. プログラムの実行	5-12
5-4-8. ActiveXコントロールを使用した簡易オシロスコーププログラムリスト	5-12

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	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によるアナログ入力プログラミング	6-6
6-3-1. リモートI/Oによるアナログ入力プログラム概要	6-6
6-3-2. 使用機器	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コントローラとの接続)	6-13
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)	6-13
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)	6-14
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)	6-14
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)	6-15
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)	6-15
6-3-14. 動作確認(診断モニタの実行)	6-16
6-3-15. Visual Basicの起動と画面の作成	6-17
6-3-16. 画面構成	6-18
6-3-17. 各オブジェクトのプロパティ設定	6-18
6-3-18. 標準モジュールファイルの追加	6-20
6-3-19. API-CAP (W32) の処理体系	6-21
6-3-20. 変数の宣言	6-21
6-3-21. 初期化処理・入力レンジ設定・デバイス起動状態の取得と起動処理の記述	6-22
6-3-22. 終了処理の記述	6-24
6-3-23. タイマ開始処理の記述	6-25
6-3-24. タイマ停止処理の記述	6-25
6-3-25. アナログ入力および表示処理の記述	6-26
6-3-26. プログラムの実行イメージ	6-26
6-4. アナログ入出力カード(ADA16-8/2(CB)L)とのプログラム比較	6-27
6-5. API-CAP (W32) アナログ入出力ドライバ提供関数一覧	6-29

付録(各種開発ツール紹介) 付録-1

①計測システム開発用ActiveXコンポーネント集 ACX-PAC (W32)	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

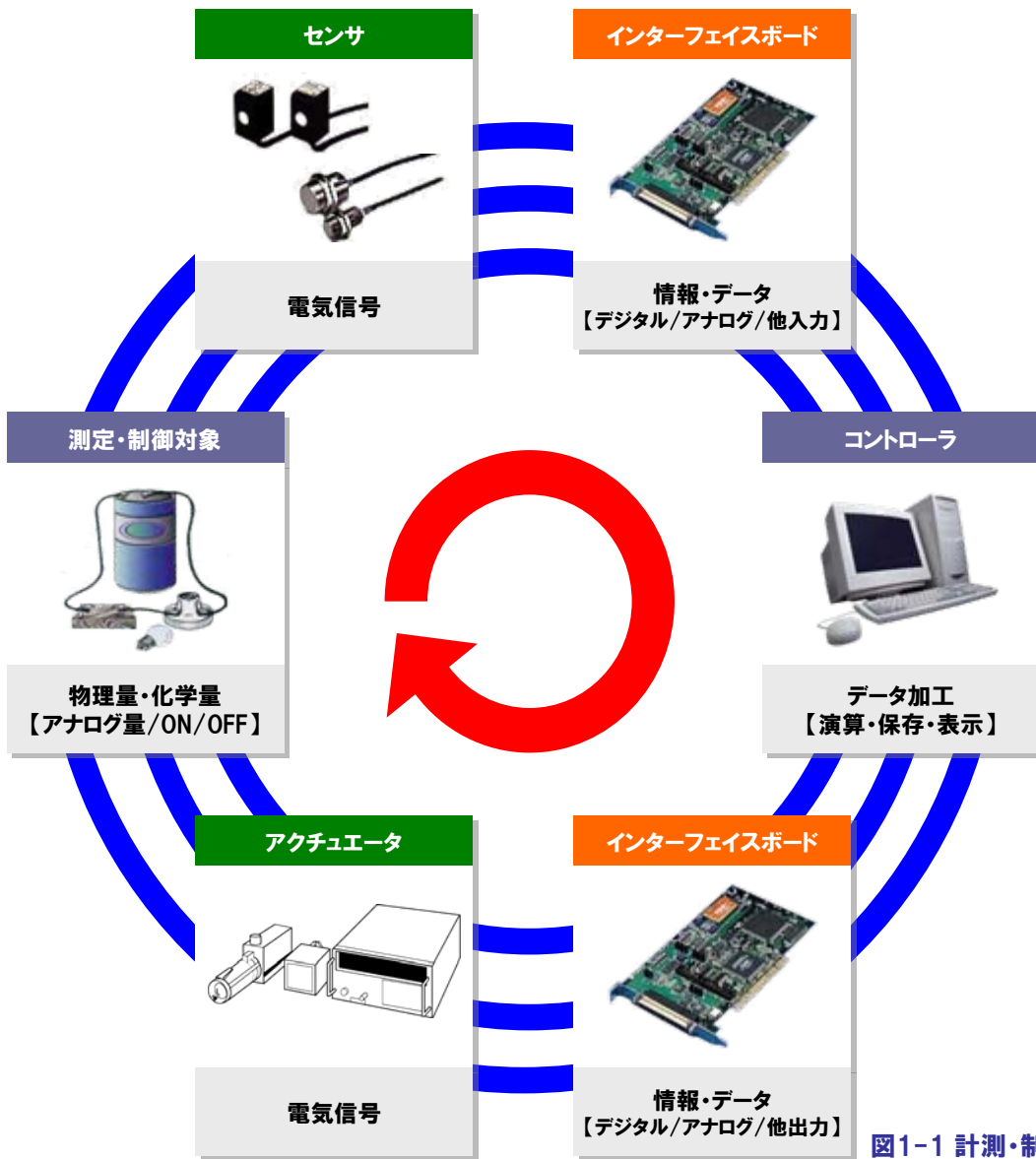


図1-1 計測・制御システムの流れ

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 °C	±0.3°C	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 ²	±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

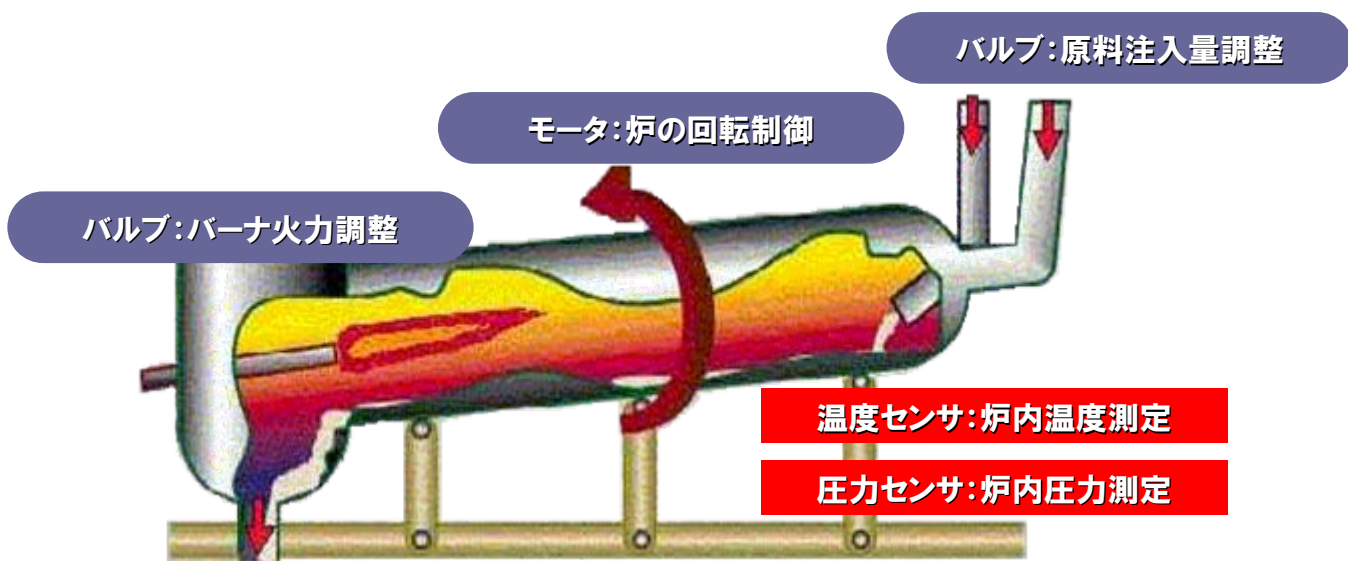
1-2. アクチュエータの主な種類と入力信号例

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。

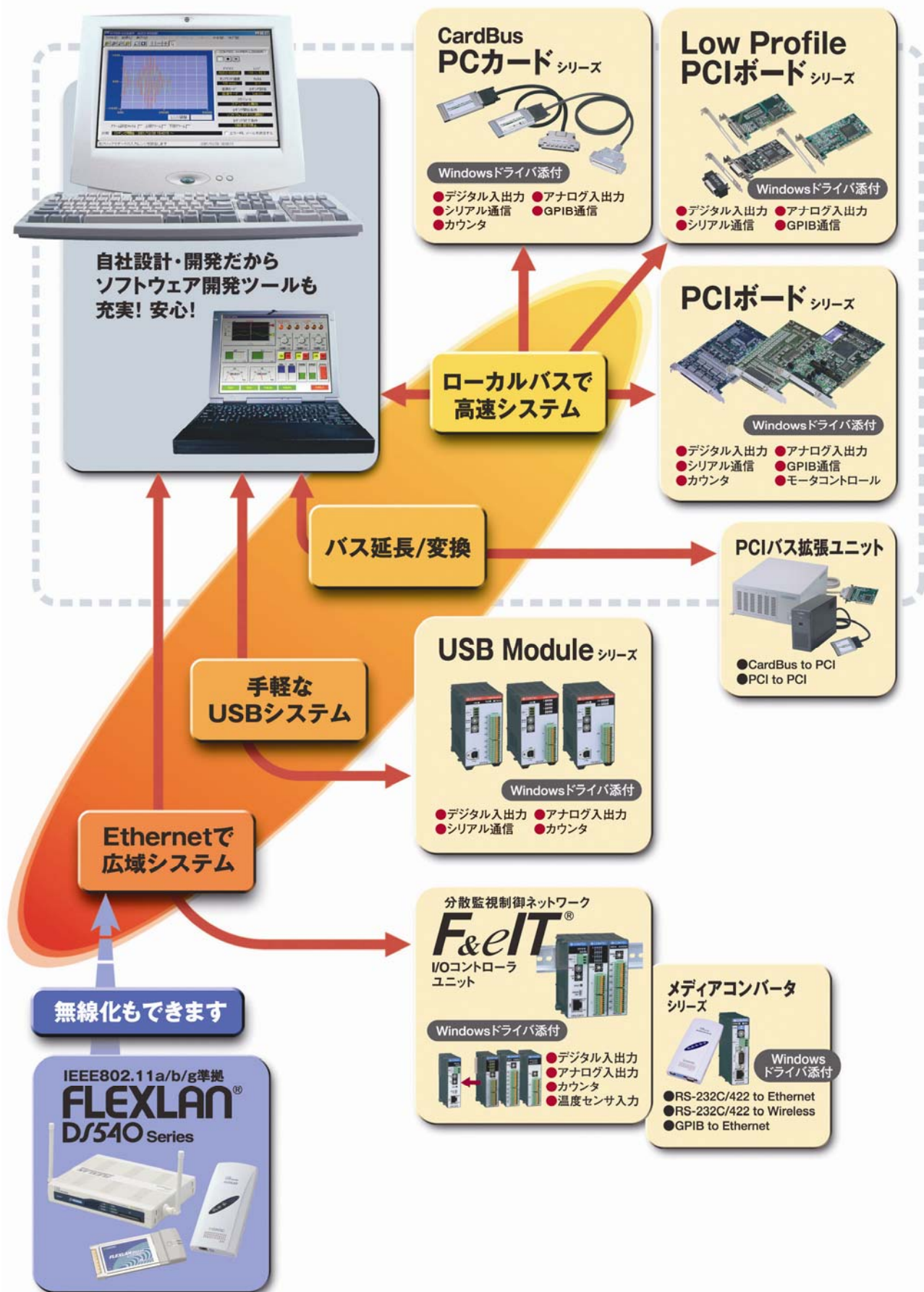
USBはUSBハブを介して、パソコンなどのホストに最大127台の機器を接続できるため、拡張バススロット数に制限されない機器の増設が可能です。データ転送速度は、12Mbps (フルスピードモード) と1.5Mbps (ロースピードモード)、最新の「USB2.0」では480Mbps (ハイスピードモード) に対応しています。

●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。

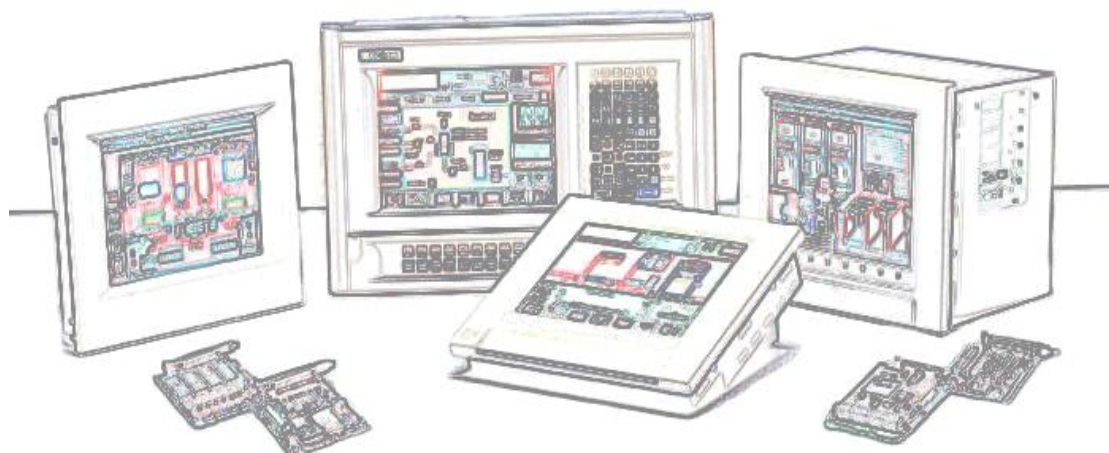
1-6. コンテックのパソコン計測・制御バリエーション

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

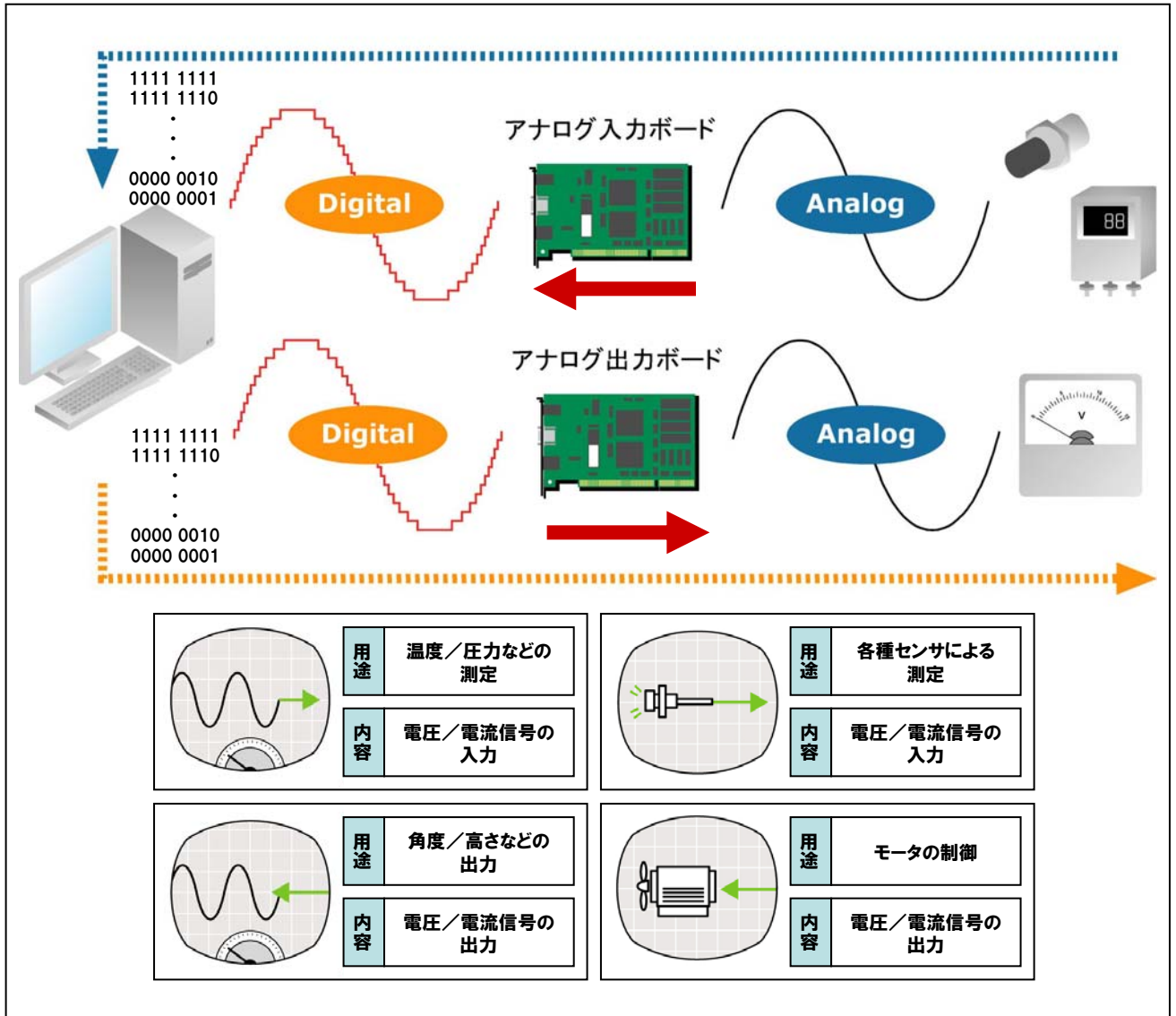
アナログ入出力の基礎知識



第2章 アナログ入出力の基礎知識

2-1. アナログ入出力とは

第1章で説明したとおり、自然界を取り巻く温度・圧力・流量などを計測するセンサからの信号はアナログ信号、また制御するためのアクチュエータの多くはアナログ信号で動作するものです。一方パソコンで扱えるのはデジタル信号のみですから、パソコンを利用してセンサからの信号を入力したり、アクチュエータへ信号を出力したりする場合は、アナログ信号とパソコンで扱えるデジタル信号との橋渡しを行うものが必要となります。それが、アナログ入出力インターフェイスです。



2-2. アナログ入出力ボード/カードの分類

● アナログ入力ボード/カード (A/D変換)

外部装置からのアナログ信号をデジタル信号に変換し、パソコンに入力するためのボード/カード。

● アナログ出力ボード/カード (D/A変換)

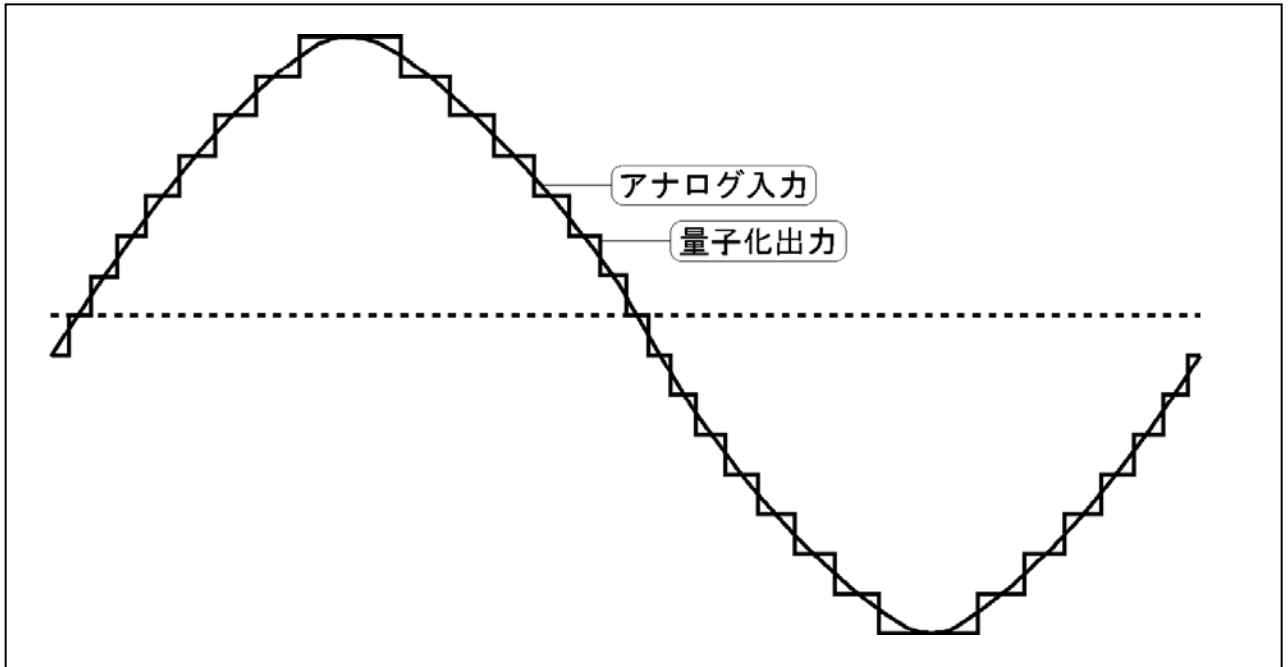
パソコンからのデジタル信号をアナログ信号に変換し、外部装置へ出力するためのボード/カード。

● アナログ入出力ボード/カード (A/D & D/A変換)

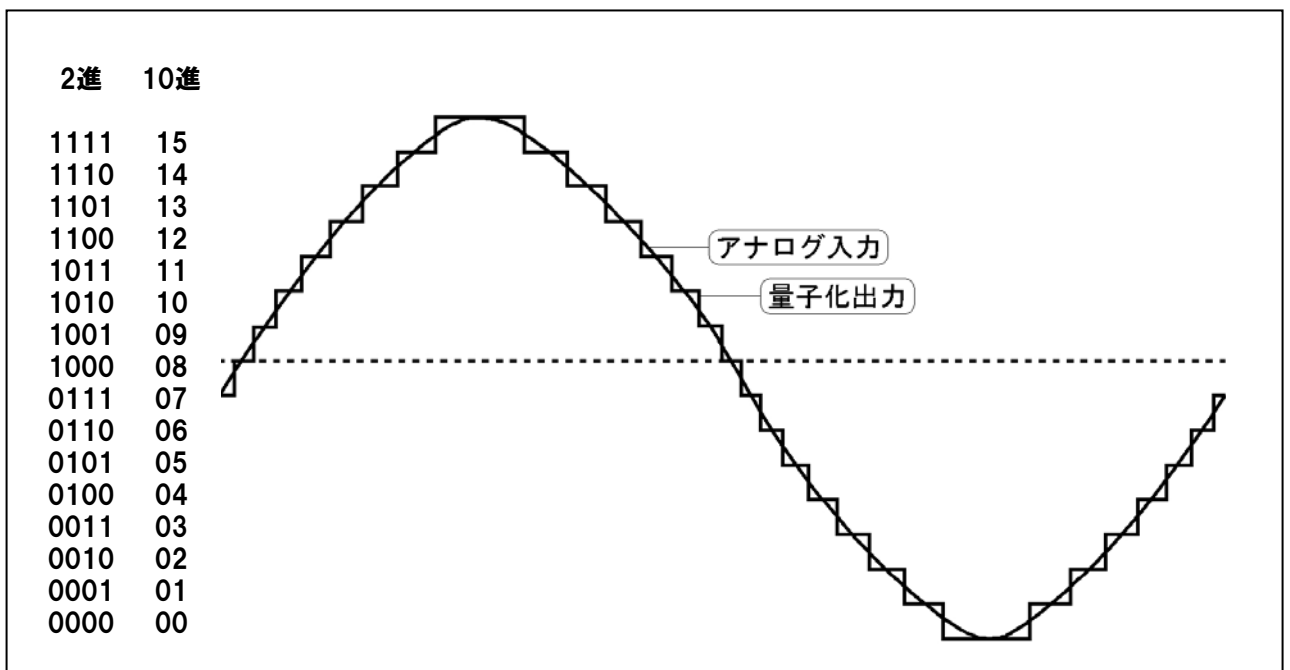
A/D変換とD/A変換の機能を合わせ持ったボード/カード。

2-3.アナログ信号のデジタル化、デジタル信号のアナログ化

パソコンに外部のアナログ量を入力しようとするとき、厳密にデジタル表現しようとする、対応するデジタル量は無限の桁数が必要となります。これは、限りある桁数しか扱えないパソコンにとってはもちろん、変換器を構成する回路技術の面から見ても不可能です。これを解決する最も有効な方法は、四捨五入や切捨て・切上げなどによって許容できる範囲内に収めてしまう方法、つまり、ある範囲内の量をその代表値で置き換えてしまうのです。これを量子化と呼んでいます。実線で表したアナログ量を量子化すると、階段状の線ようになります。これによってアナログ信号を有限の値のどれかで表すことが可能となるのです。この技術は皆さんの身近なところで活躍しています。例えば携帯電話。皆さんの音声(アナログ)をデジタルに変換して通話を行っています。



例えば階段の1段階を1として10進数で表し、この10進数をさらに2進数に置き換えると、下図のようになります。こうすればアナログ量を4ビットでデジタル化したことになり、これがアナログ量を量子化する基本的な考え方です。

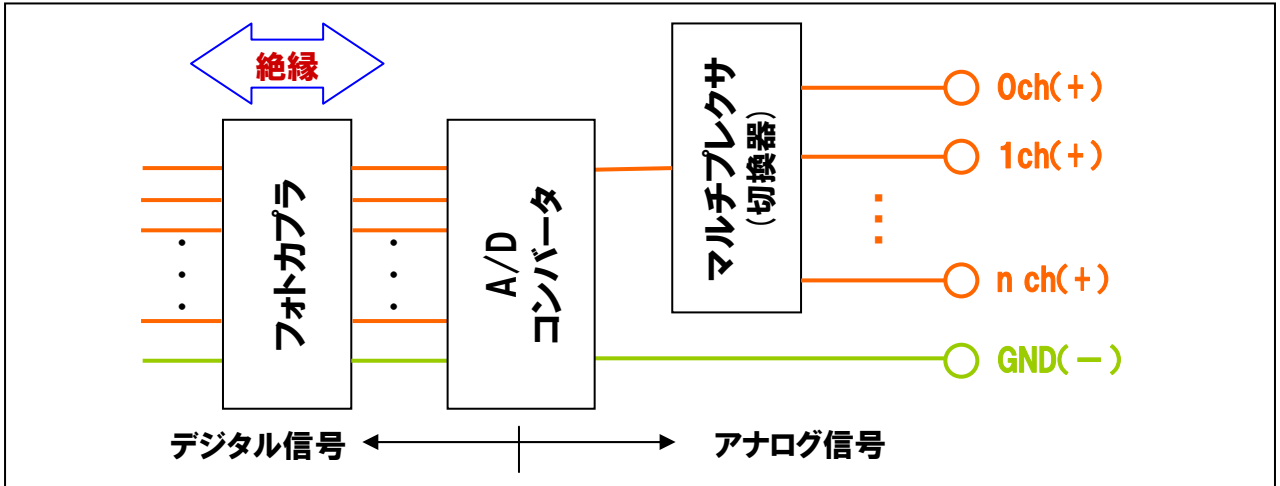


2-4. アナログ入出力ボード/カードの絶縁タイプ

アナログ入出力ボード/カードは、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。ここでは、『絶縁型』に属する2つのタイプについて、その特長を解説します。なお、『非絶縁型』については、2-4-1.中の図において、絶縁素子が介在しないタイプを示します。

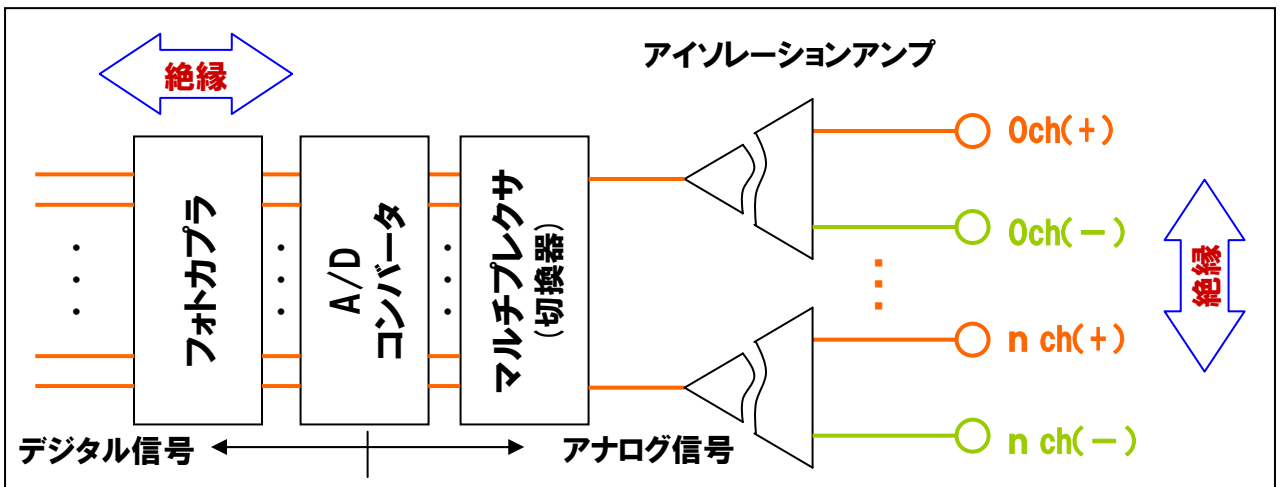
2-4-1.バス絶縁型

フォトカプラによってパソコンと外部入出力回路を絶縁しています。電気的外乱の侵入を防ぐことができますので、配線上にノイズが発生しやすく、パソコンの誤動作や破損が心配される場合にも安心して使用することができます。



2-4-2.独立絶縁 (チャンネル間絶縁) 型

バス絶縁に加えて、フォトカプラやアイソレーションアンプによって各入出力チャンネル間を絶縁しています。各チャンネル間の干渉を防ぐことができ、各チャンネル接続する機器それぞれでグランドレベルが異なる場合でも正確にサンプリングできます。

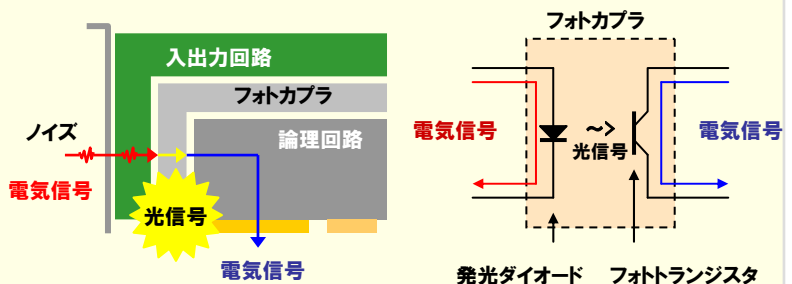


TOPICS

「フォトカプラとは・・・」

フォトカプラとは、発光ダイオードとフォトランジスタを1対にした素子です。

発光ダイオードは電流 (10mA) 程度が流れると光を出します。この光をフォトランジスタが受光すると、フォトランジスタがON状態になり、電流が流れます。この光信号の部分で、外部と電氣的に絶縁されます。



2-5. 用語解説

カタログやデータシートを参考にして、アナログ入出力ボード/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アナログ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、システムに最適なアナログ入出力ボード/カードを選定できるようになります。

2-5-1. 入出力チャンネル数

入出力チャンネルとは、1枚のボード/カードが持つ入力または出力可能な信号数を意味します。つまり、接続できるセンサ（信号源）やアクチュエータ（制御対象）の数を表しています。

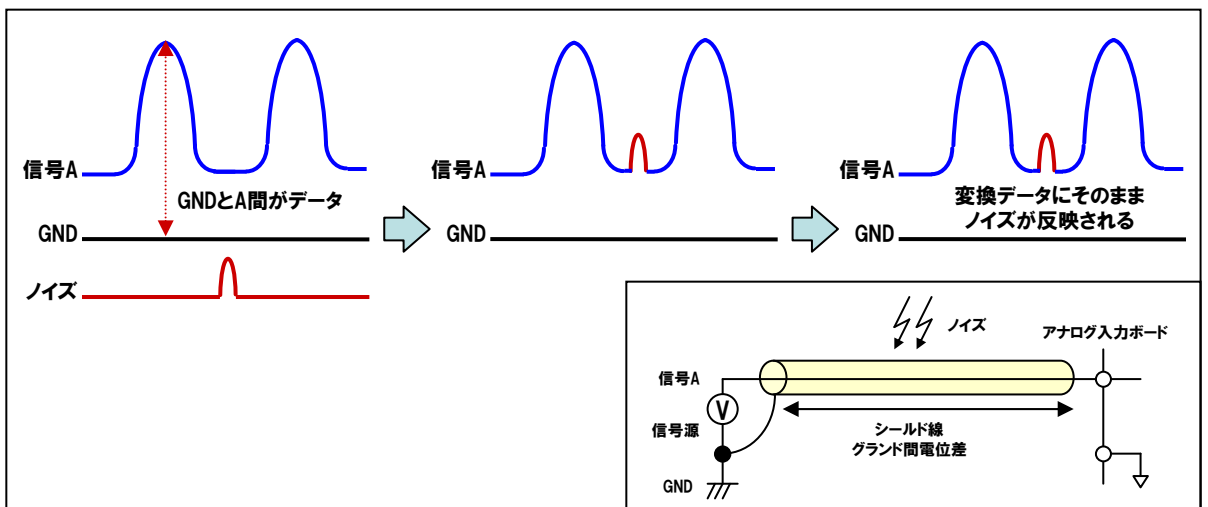
カタログなどの仕様欄を見ると、シングルエンド**ch、差動**chという表記がされている場合があります。

ここで、シングルエンド入力と差動入力について解説をしておきます。

① シングルエンド入力

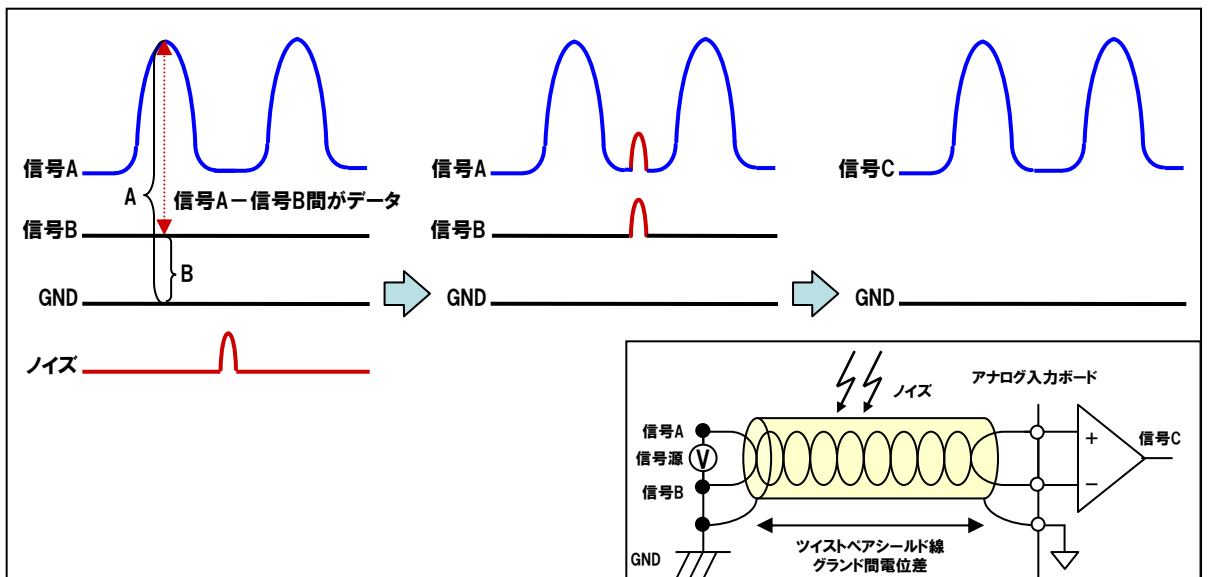
信号線とグランド線の2線で接続して、グランドからの電位差で信号源の電圧を測定する方式です。アナログ入力では最も一般的な入力方式で、1つの信号源に対して、配線が2線で済むというメリットがあります。

②で説明する差動入力と比較して、ノイズの影響を受けやすいというデメリットがあります。



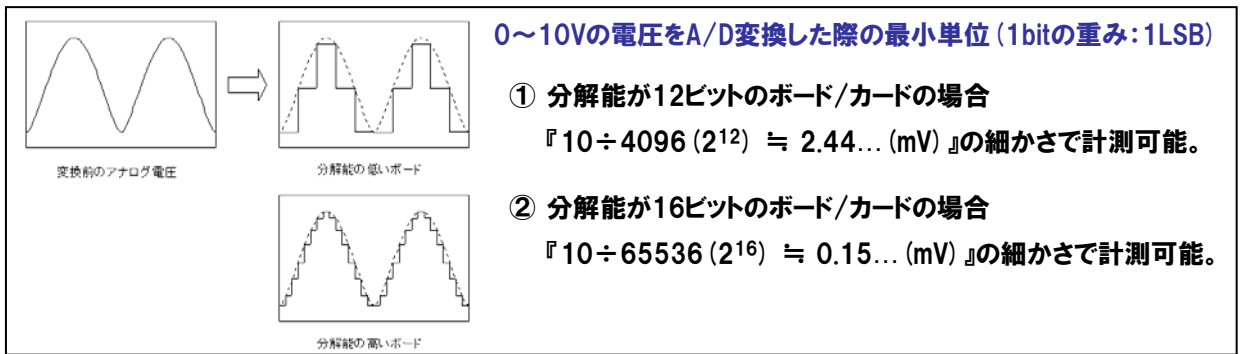
② 差動入力

2つの信号線とグランド線の合計3線で信号源の電圧を測定する方式です。グランドとA点間の電位とグランドとB点間の電位の差をとって信号源（A-B間）の電位を測定します。そのため、（A-B間）はグランドにのったノイズは相殺され、シングルエンド入力と比較して、ノイズの影響を受けにくいというメリットがあります。ただし、1つの信号源に対して、配線が3線必要となるため、シングルエンド入力と比較して、使用できるチャンネル数が半分になるといったデメリットもあります。



2-5-2.分解能

アナログ信号をどの程度の細かさでデジタル表現(近似)できるかを示します。分解能が高いほど、電圧の範囲が細かく区分されていることになり、アナログ値をより正確にデジタル値に変換することができます。



多種多様なアナログ入力ボード/カードの中で、自分に最適な分解能を持ったボード/カードを選ぶには、どのようにして考えれば良いかの例を示しましょう。例えば『0℃～100℃の温度を計測したい』とします。

例1:【1℃単位で計測したいのなら】

→ 1/100の精度が必要 → 分解能8ビット ($2^8=256$ 分割) のボード/カードで十分。

例2:【0.1℃単位で計測したいのなら】

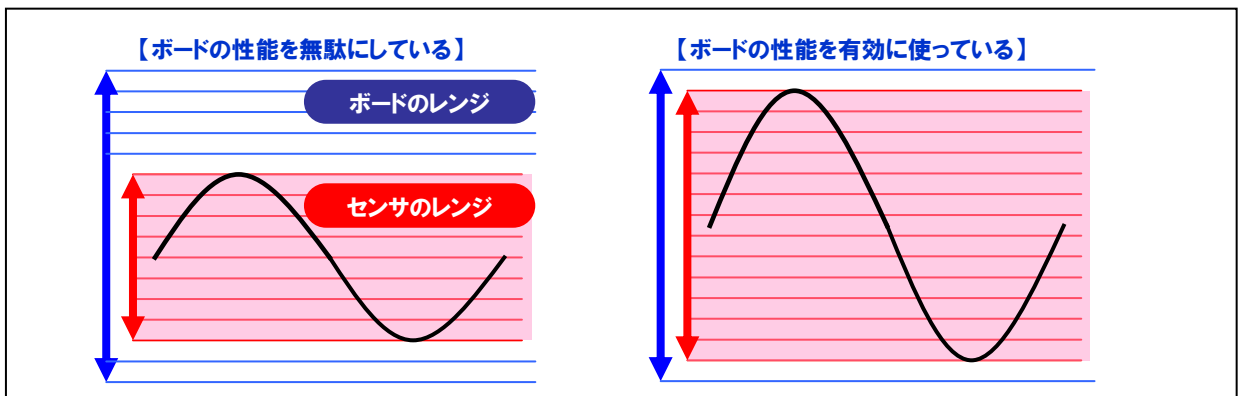
→ 1/1000の精度が必要 → 分解能12ビット ($2^{12}=4096$ 分割) のボード/カードが必要。

例3:【0.01℃単位で計測したいのなら】

→ 1/10000の精度が必要 → 分解能16ビット ($2^{16}=65536$ 分割) のボード/カードが必要。

2-5-3.入出力レンジ

入力または、出力可能なアナログ電圧・電流の範囲です。ハイポーラは双極性(-10V～+10Vレンジなど)、ユニポーラは単極性(0～+10Vレンジなど)という意味です。入出力レンジは、センサからの出力、アクチュエータへの入力と同一、もしくは少し広い範囲のレンジを持ったボード/カードを選定するのが基本です。



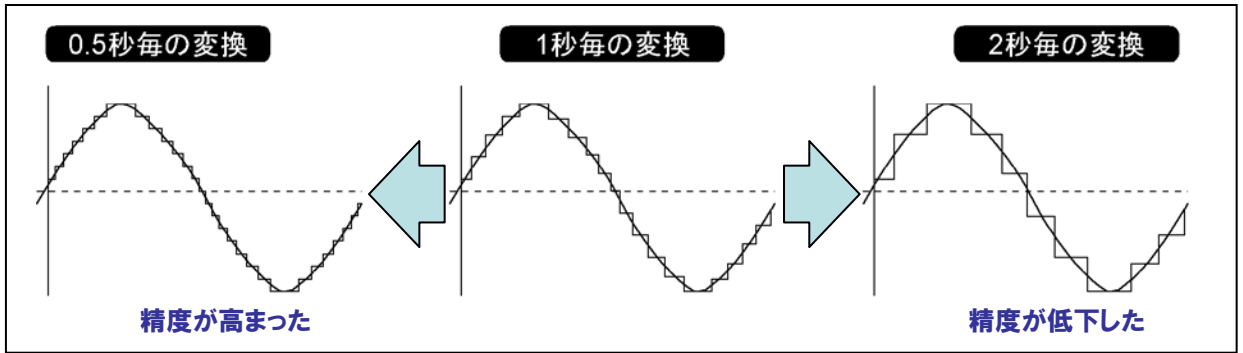
例えば、あるアナログ量を電圧の0～5Vに変換してくれるセンサを使うと仮定します。そこで、入力レンジが0～10Vと0～5Vのボードでは、どちらが有効かという点、分解能が共に12ビットとしたら、レンジが0～10Vのボードが最小分割できる電圧は、 $10 \div 4096$ で約2.44mVです。0～5Vのボードは、 $5 \div 4096$ になりますから、最小分割できる電圧は、約1.22mVです。センサは0～5Vしか出力しないのですから、入力レンジ0～5Vのボードを選んだ方が、より細かく計測することができます。

2-5-4.ゲイン

ゲインとは倍率を意味します。アナログ入力ボード/カードの中には、入力信号を増幅する機能を搭載したものがああります。例えば、外部信号が0～2.5Vの場合、アナログ入力ボード/カードの入力レンジが0～10Vと仮定すると、そのままの状態に変換するより、外部信号(入力される信号)を4倍に増幅して、0～10Vの信号として変換した方が高精度で計測することができます。

2-5-5.変換速度 (サンプリング周期)

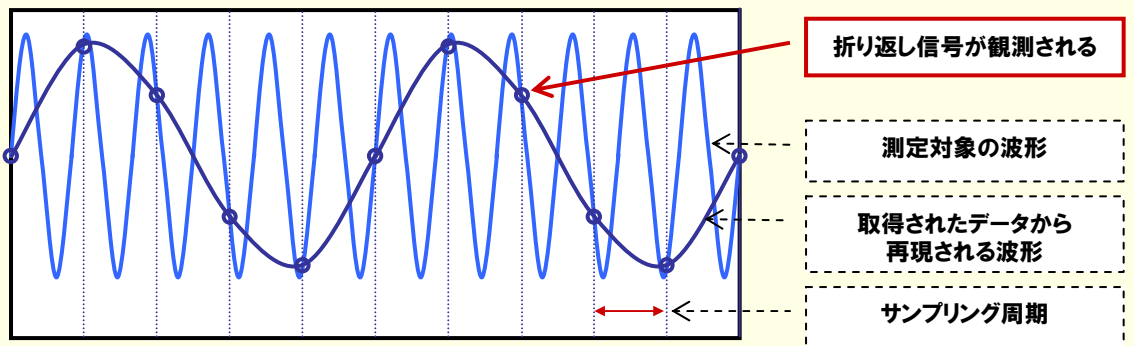
アナログ信号をどの程度の時間間隔の細かさでデジタル変換できるか。または、デジタルデータをどの程度の時間間隔でアナログ出力できるかを表します。変換速度が速いほど、再現性の高い変換が可能となります。



TOPICS

「サンプリングの定理」

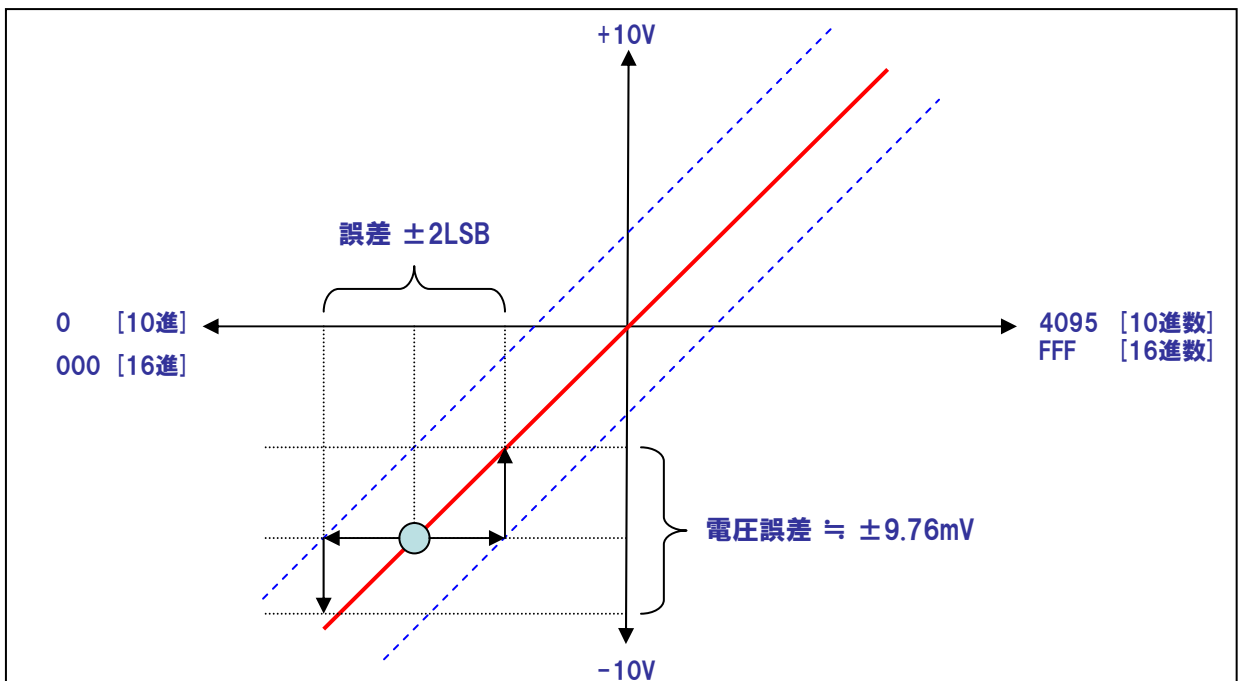
入力/測定周波数の2倍以上のサンプリング周期でサンプリングしないと正確な波形測定ができません。



2-5-6.変換精度

A/D変換またはD/A変換の際に生じる誤差範囲です。誤差を1LSB単位で表現します。例えば、分解能12ビットのA/D変換ボードで、入力レンジ±10Vの設定とすると、分解できる最小単位は、 $20 \div 4096 \approx 4.88\text{mV}$ (1LSB) となります。そのA/D変換ボードの変換精度が±2LSBと表記されている場合、 $4.88 \times 2 \approx \pm 9.76\text{mV}$ 程度の誤差が生じる可能性があるという意味になります。

※LSB:『Least Significant Bit』の略で、バイナリデータ (2進数データ) の最下位ビットを表します。



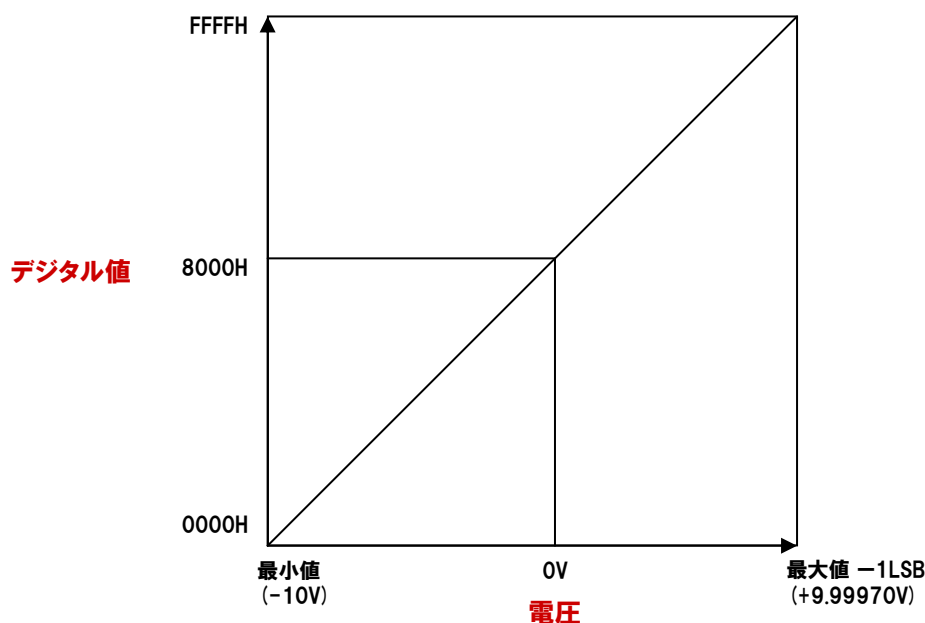
2-5-7. バイナリデータ (2進数データ) と電圧値の関係 (分解能16ビットの場合)

本書で使用しているアナログ入出力カードADA16-8/2 (CB) Lの分解能は16ビットです。アナログ入力の場合、カードから入力されるA/D変換データは、バイナリ16桁 (16進数=4桁) になり、アナログ出力の場合にカードにセットするD/A変換データはバイナリ16桁 (16進数=4桁) で行います。これらバイナリデータと電圧値との関係を見てみましょう。

【表2-1.】の『FFFF』の欄に注目してください。『0000』のときの『-10』に対して、電圧は『9.9997・・・』です。分解能16ビットのボード/カードの場合、-10V～0Vまでの電圧は-10Vを「0000H」とし、0Vを「8000H」とした32768通りのデータとして表現できるのに対し、0V～+10Vは、「8000H」から最大値「FFFFH」までの、32767通りしか表現できないのです。このため最大値は「+10V-1LSB」ということになります。これらの関係は、どのアナログ入出力ボード/カードを使用しても共通でいえる事柄です。例えば、アナログ出力をする場合、外部に信号を出力する際には、分解能16ビットの場合は『FFFF』をボード/カードにセットするわけですが、実際に出力できる最大電圧は、『+10V-1LSB』までしか出力できません。

電圧の意味	アナログ電圧 (V)	DATA (16進数)	2進数表記 (バイナリデータ表記)	
			MSB	LSB
FSR-1LSB	9.99970	FFFF	1111 1111 1111 1111	
FSR-2LSB	9.99938	FFFE	1111 1111 1111 1110	
:	:	:	:	:
+1LSB	0.00030	8001	1000 0000 0000 0001	
	0.00000	8000	1000 0000 0000 0000	
:	:	:	:	:
-1/2FSR+1LSB	-9.99970	0001	0000 0000 0000 0001	
-1/2FSR	-10.00000	0000	0000 0000 0000 0000	

表2-1



TOPICS

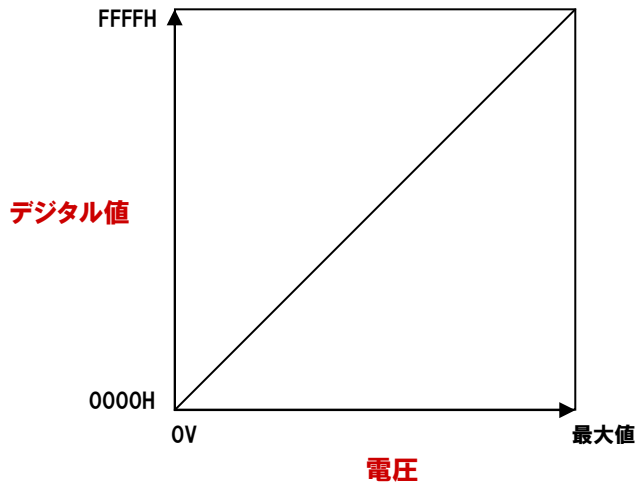
「略語の意味」

- ①LSB :Least Significant Bitの略です。バイナリデータ (2進数データ) の最下位ビットを表します。
- ②MSB :Most Significant Bitの略です。バイナリデータ (2進数データ) の最上位ビットを表します。
- ③FSR :Full Scale Rangeの略です。±10Vレンジの場合は『20』がFSRとなります。

アナログをデジタル値で表現するときに、いくつかの表現方法があります。これはハードウェアの仕様検討にはあまり関係しませんが、プログラムをする場合に必要な知識となりますので、解説をしておきます。

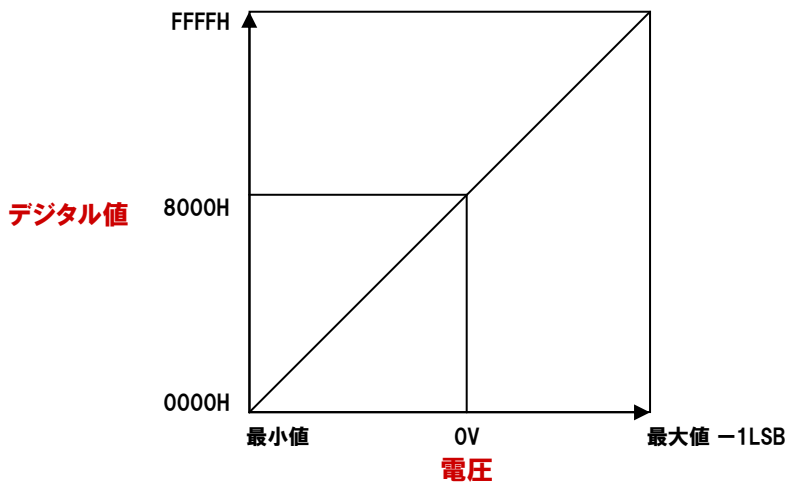
① ストレート・バイナリ

0Vをデジタル値の"0"とし、電圧の増加に従いデジタル値も増加します。ユニポーラ(単極)型で用いられます。



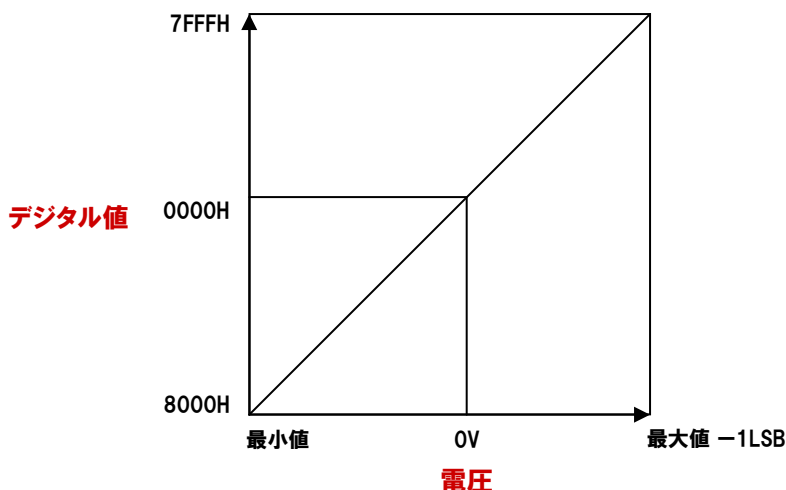
② オフセット・バイナリ

負電圧の最大値をデジタル値の"0"とします。電圧の0Vをデジタル値の中間値、正電圧の最大値をデジタル値の最大値とします。バイポーラ(双極)型で用いられます。前項の[表2-1]の関係がこれにあてはまります。



③ コンプリメント・バイナリ(2の補数)

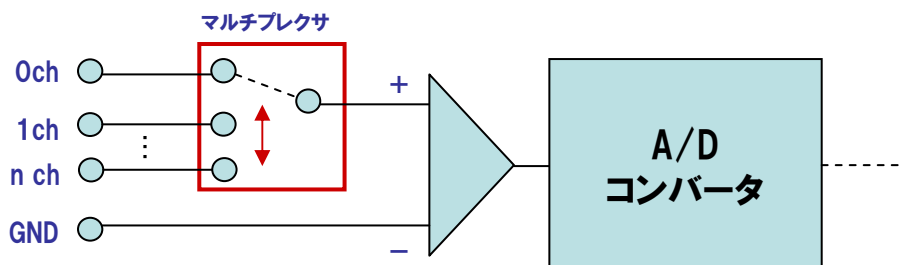
オフセット・バイナリコードの最上位ビットを反転させて得られます。2の補数表現は、コンピュータ上の演算に使い易いコードです。バイポーラ(双極)型にこのデータ形式のものもあります。



2-5-8. サンプルング方式

複数チャンネルのサンプルングを行う場合には、マルチプレクサ (切換器) にマルチプレクサ方式と、同時サンプルング方式があります。マルチプレクサ方式は、マルチプレクサの切換えによるサンプルングで、複数チャンネルを同時変換することが出来ません (チャンネル切換え時間が必要なため)。同時サンプルング方式は、チャンネル毎にA/Dコンバータが搭載されたタイプと、サンプル/ホールドアンプが搭載されたタイプがあり、どちらも複数チャンネルの同時変換が可能となっています。

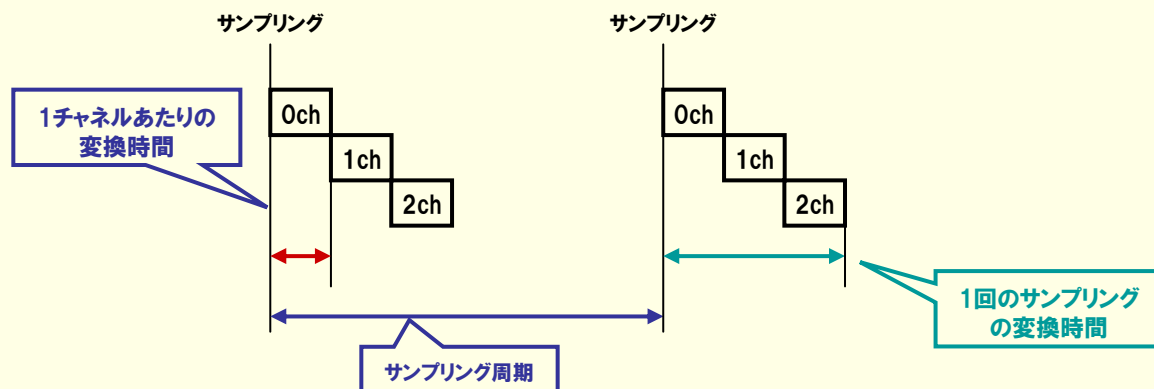
マルチプレクサ方式のイメージ (弊社アナログ入力ボード例)



TOPICS:「変換速度とチャンネル数の関係」

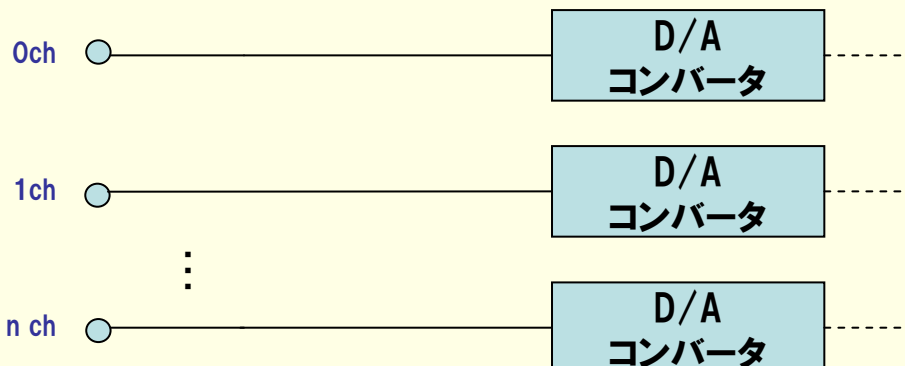
マルチプレクサ方式によるアナログ入力チャンネル切替方式の場合、複数チャンネルのサンプルングを行う際、設定可能なサンプルング周期は次の関係を保つ必要があります。

→ 『変換速度 × チャンネル数 ≤ サンプルング周期』



TOPICS:「同時出力方式のイメージ (弊社アナログ出力ボード/カード)」

弊社製のアナログ出力ボード/カードは、チャンネル毎にD/Aコンバータを搭載しており、全てのチャンネル出力電圧を同時に更新する設定が可能です。



2-5-9.クロック

アナログ入力ボード/カードの変換動作をどのタイミングに同期させることができるかを示します。
サンプリングの周期を決定するサンプリングクロックには、主に以下の方式があります。

①内部クロック

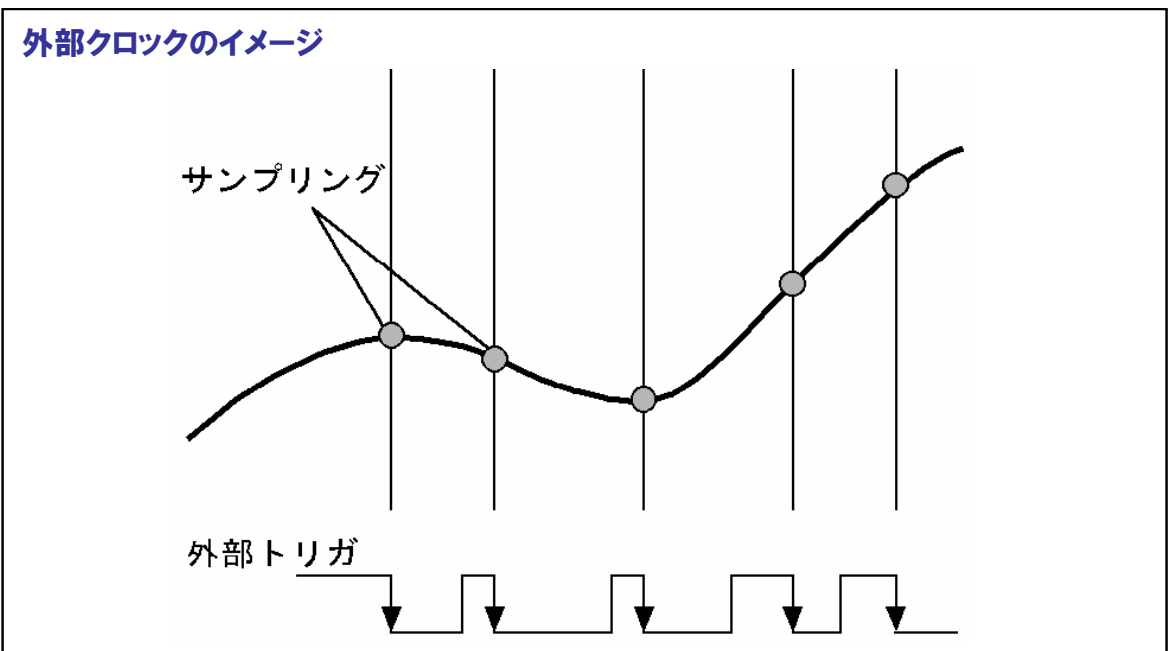
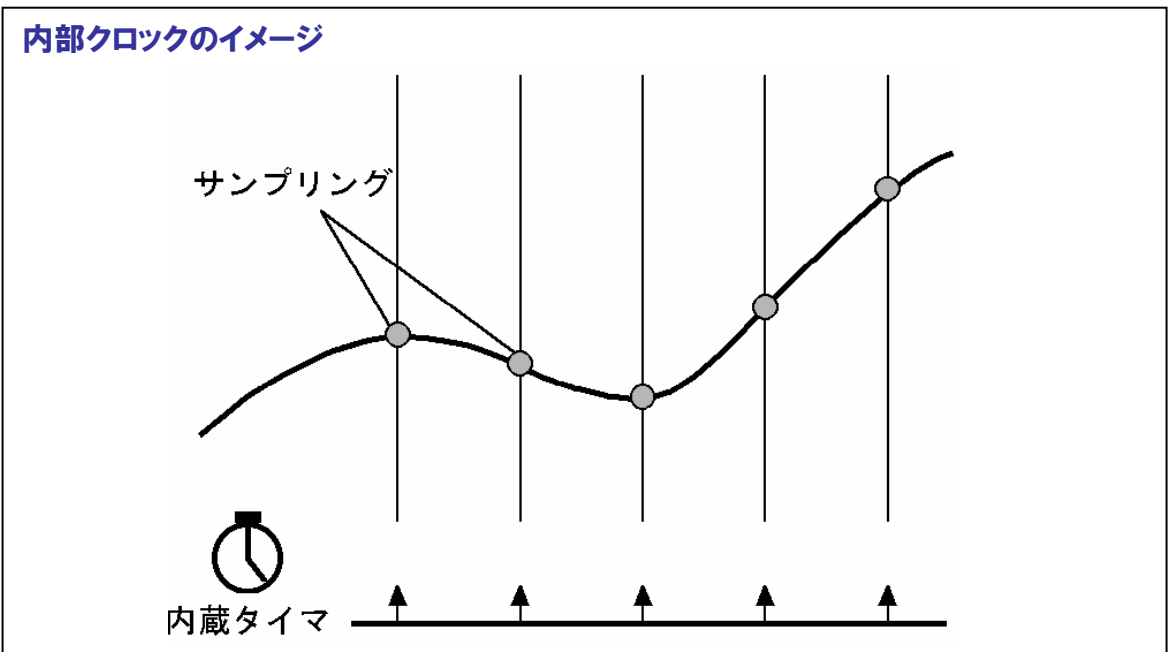
ボード/カードに周期の設定が可能なタイマ素子を搭載し、これをクロック源として周期的な変換を行う方法です。正確かつ短周期での時系列処理に有効です。

②外部クロック

外部クロック入力端子を搭載したボード/カードで使用可能です。外部から入力されるパルス信号などに同期して、変換を行います。外部装置との同期処理などに有効です。

③ソフトウェアクロック

パソコンのシステムタイマに同期してソフトウェア上からスタートコマンドを送信し、周期的な変換を行う方法です。ただし、Visual Basicのタイマコントロールなどは誤差が大きいため、高速で正確な周期が必要なシステムには向きません。

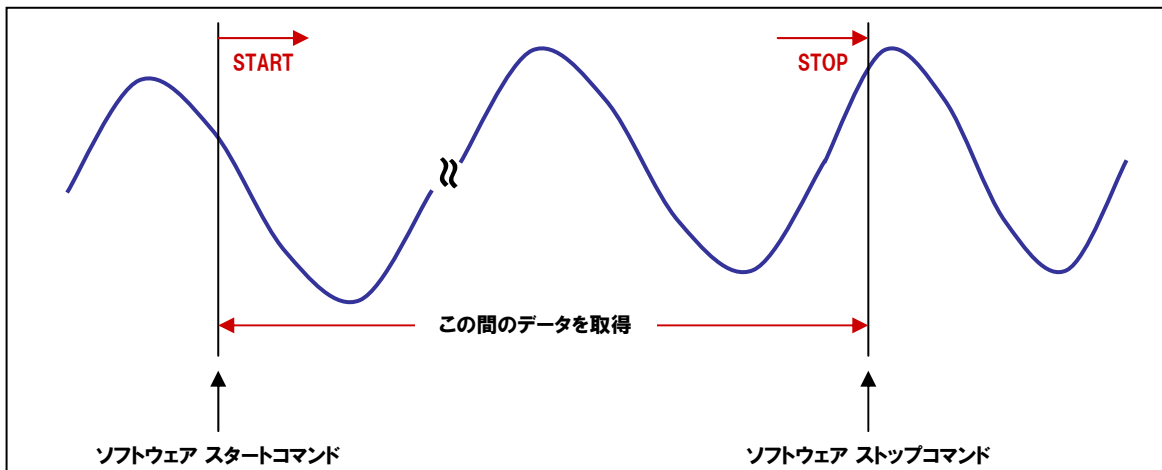


2-5-10.トリガ

変換の開始と停止をどのタイミングで実行するかはの要因です。開始、停止それぞれ独立して設定可能です。主なトリガは以下のとおりです。

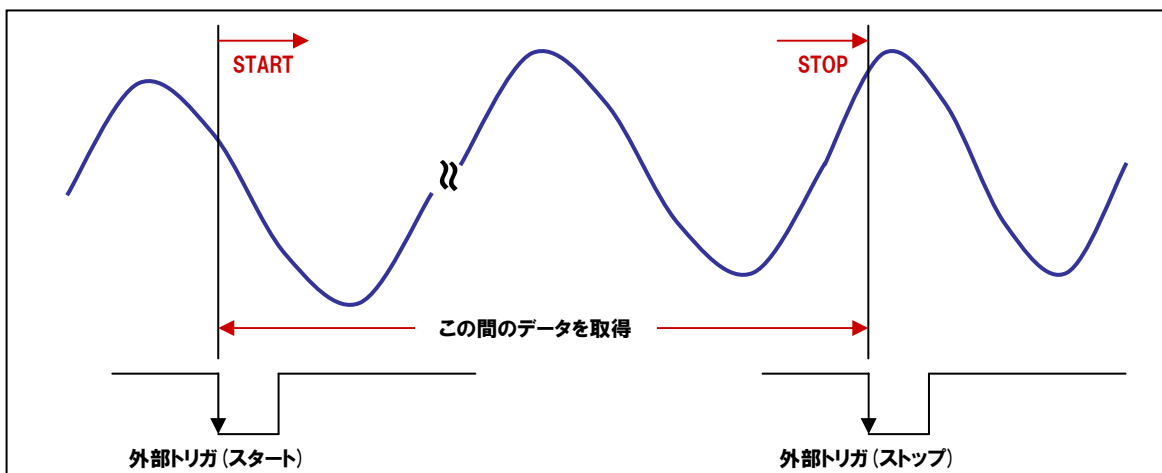
①ソフトウェアトリガ

変換動作の開始/停止をソフトウェアからのコマンドで制御する方法です。



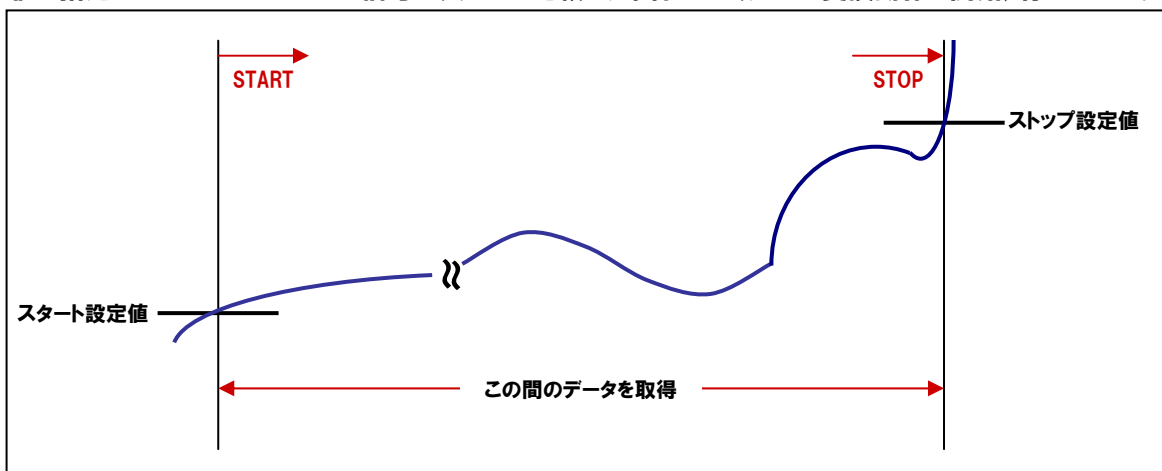
②外部トリガ

変換動作の開始/停止を外部信号 (デジタル信号) で制御する方法です。あらかじめ設定したエッジの方向 (立ち上がり、立ち下がり) の外部制御信号が入力されると変換動作を開始/停止します。



③レベル比較 (変換データ比較) トリガ

変換動作の開始/停止の制御を指定チャンネルの信号変化で行います。あらかじめ設定した比較レベルの値と指定したチャンネルのアナログ信号の大きさを比較し、条件に一致すると変換動作を開始/停止します。

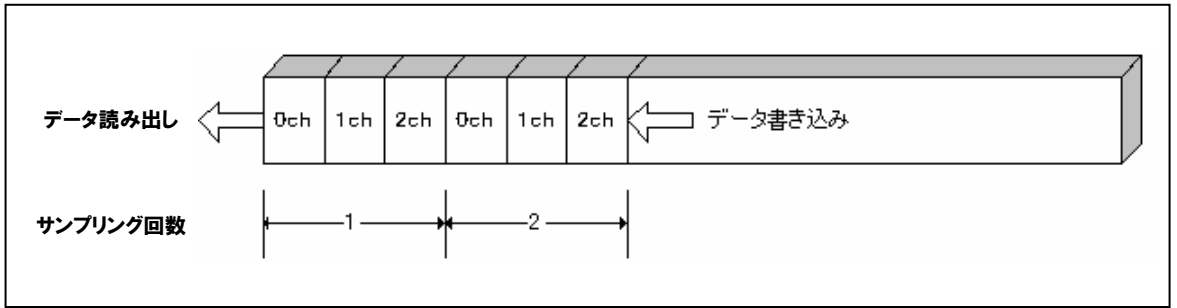


2-5-11.バッファメモリ

バッファメモリとは変換データを一時的に保管している場所のことです。高速かつ高機能なアナログ入出力処理が可能だけでなく、パソコン側の負荷を大幅に軽減することができます。バッファメモリは用途に応じて、FIFO形式とRING（リング）形式があります。

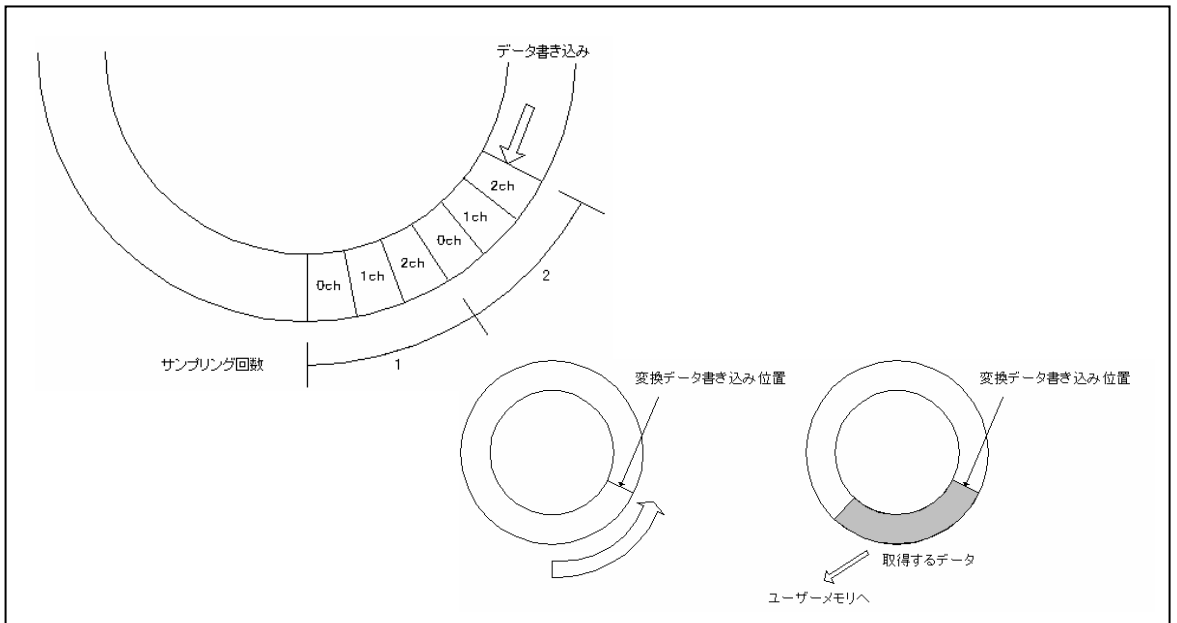
①FIFO形式

FIFO (First In First Out) 形式では、変換されたデータが前詰めでバッファメモリに格納され、バッファメモリに書き込まれた先頭から古い順に読み出すことができます。読み出した変換データはメモリ内部から順次送り出され、つねにバッファメモリに残っている一番古い変換データを読むことができます。FIFOメモリの容量を超えるデータは書き込まれず破棄され、また、一度読み出したデータはバッファメモリ上から破棄されます。



②RING（リング）形式

RING（リング）形式では、バッファメモリ内部の格納領域がリング状に構成されています。変換データは順次書き込まれていき、メモリ容量を超えて格納を続けると前の変換データが格納されている領域に上書きされます。通常の状態ではデータ取得を行わず、何かの事象で変換動作が停止した付近のデータを取得するような場合、RINGメモリを使用します。RING（リング）形式の場合、一度取り込んだデータは、次に上書きされるまでは、何度でも取り込み可能です。

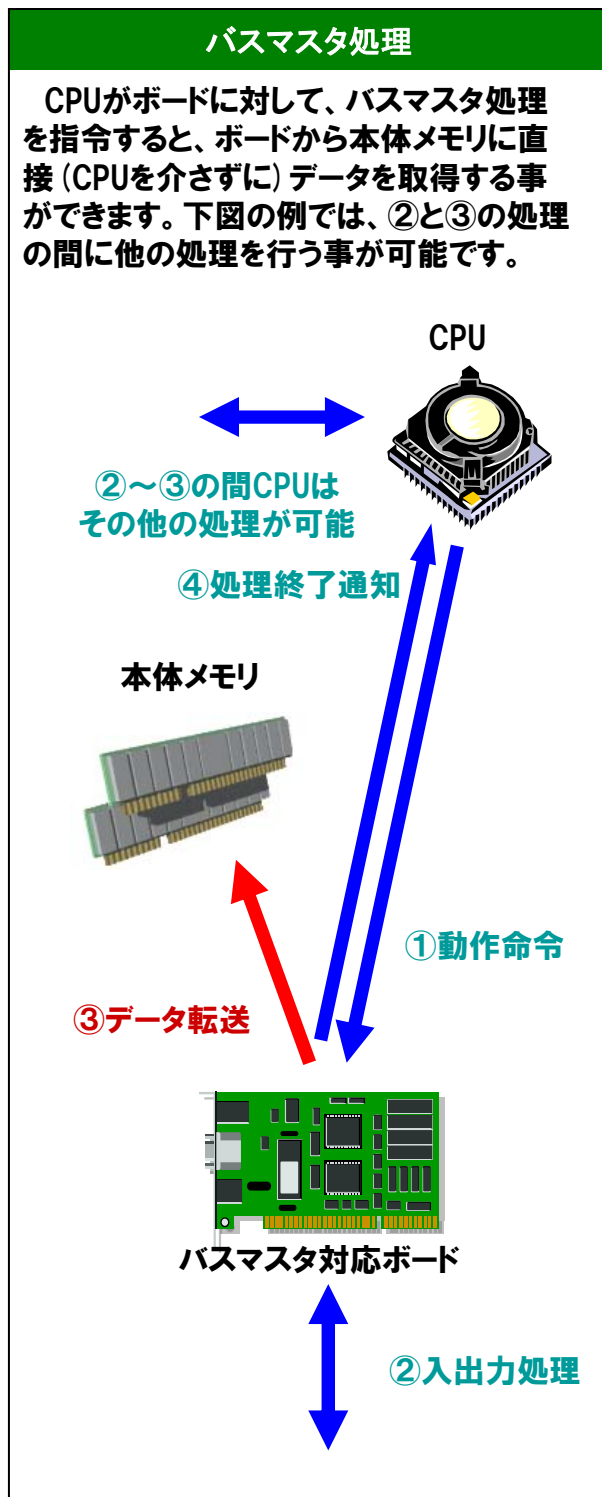
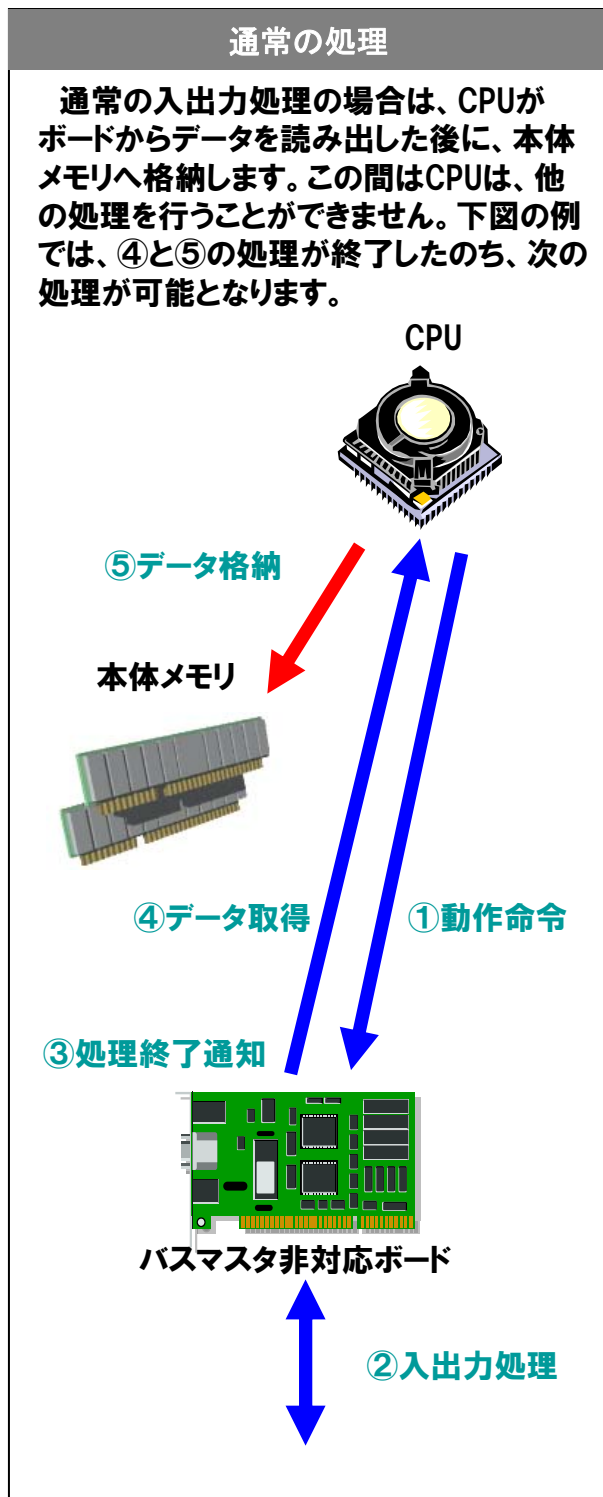


2-5-12. バスマスタ転送機能

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/カードです。

■特長

- [1] パソコンのCPUに負荷をかけることなく、ボード/カ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。



2-5-13.I/OポートとI/Oポートアドレス

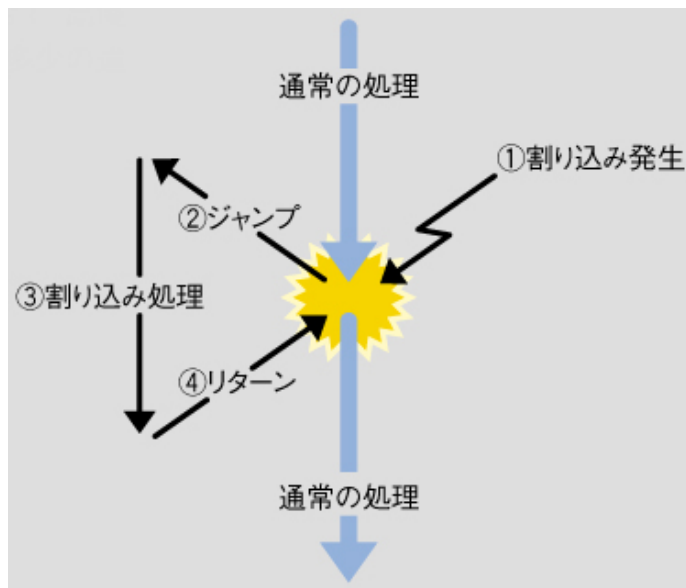
一般にパソコンの入出力命令によってコントロールされる機器 (I/Oデバイス) は、割り当てられた『I/Oポートアドレス』に対して入出力命令を実行し、動作させます。パソコンの内部で、CPUが接続された周辺機器とデータのやり取りを行うための窓口を『I/Oポート』、それを管理・識別するために割り当てられる番号を『I/Oポートアドレス』と表現します。現在主流のPC/AT互換機は、メモリ領域『0000h~FFFFh』64KBのアドレス空間が使用されます。『0000h~00FFh』など特定のI/Oアドレスは、システムによって使用済みもしくは予約済みとなっており、ユーザーでの使用が禁止されています。そのため、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、『2ⁿ』が一般的です。

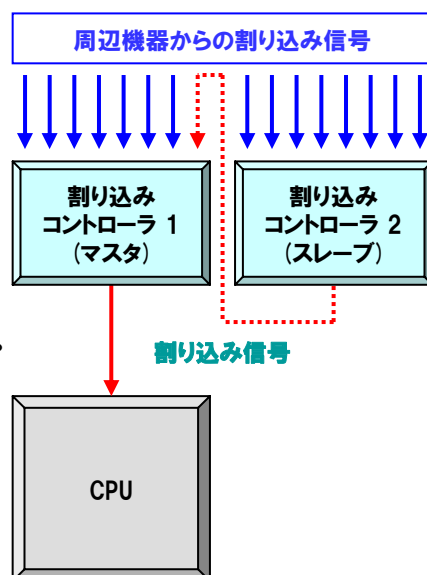
2-5-14.割り込み

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。



2-5-15.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、パソコンの電源電圧の低下をまねき、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。

例：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・・・

$1.2(A) \times 2(枚) = 2.4A \rightarrow$ 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・・・

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

2-5-16.参考資料：ノイズの種類とその対策

ノイズには大きく分けて、次の2種類が存在します。電気実験と異なり、現場にはさまざまなノイズが存在し、理論どおりにいかないケースが多々あります。このような場合、精度を狂わせる原因の多くがノイズに起因しています。

①外来ノイズ

- (a) 信号伝送ラインの外部から空中伝搬により飛来するノイズ。
- (b) モータなどの動力系機器配線との混在、近辺を經由する配線から混入するノイズ。
- (c) 動力系機器と計測系機器との電源の共用、アースの共用による回り込みから発生するノイズ。

②内部ノイズ

- (a) アナログ入出力回路の接続に起因するノイズ。
- (b) 装置間のグラウンド電位差によるオフセット電圧およびノイズ。
- (c) 配線材料に起因するクロストーク、輻射ノイズ。

対策

特に計測を行う場合の原則は、測定対象に影響を及ぼさないことです。そのためには、インピーダンスやグラウンドレベルなどのマッチングに対する配慮が必要となります。慣れれば難しいことではありませんが、配慮を怠った場合の影響は大きいと言えます。下表はノイズ対策方法の一例です。

	対策方法	具体例	効果	留意点
ハード	信号レベルの強化	微弱信号の測定点での増幅	ノイズ全般の除去	パソコン側では効果小
	配線方式による除去	シールドケーブルの使用	飛来ノイズの除去	
		ツイストペアケーブルの使用	クロストークの除去	
	配線方法による除去	計測・制御系と動力系との配線分離 (電源、GND、配管)	飛来ノイズの除去	
	フィルタ回路挿入による除去	EMIフィルタ・CRフィルタの挿入	ノイズ全般の除去	ノイズ周波数が特定できるときに有効
	入出力形式による除去	差動入出力による接続	飛来ノイズの除去	同相ノイズに対してのみ有効
	アースの接続による除去	各装置をアースに接続	装置間電位差の除去	アースによっては逆にノイズ源となる
ソフト	平均化演算による除去 (複数読み出し)	移動平均によるノイズの平滑化	高周波ノイズの除去	変化に対する応答性が悪くなる
		ブロック平均によるノイズの除去	高周波ノイズの除去	サンプリングレートが低くなる
	ソフトフィルタによる除去	フィルタ関数によるノイズの除去	高周波ノイズの除去	リアルタイムな処理には不向き

第3章

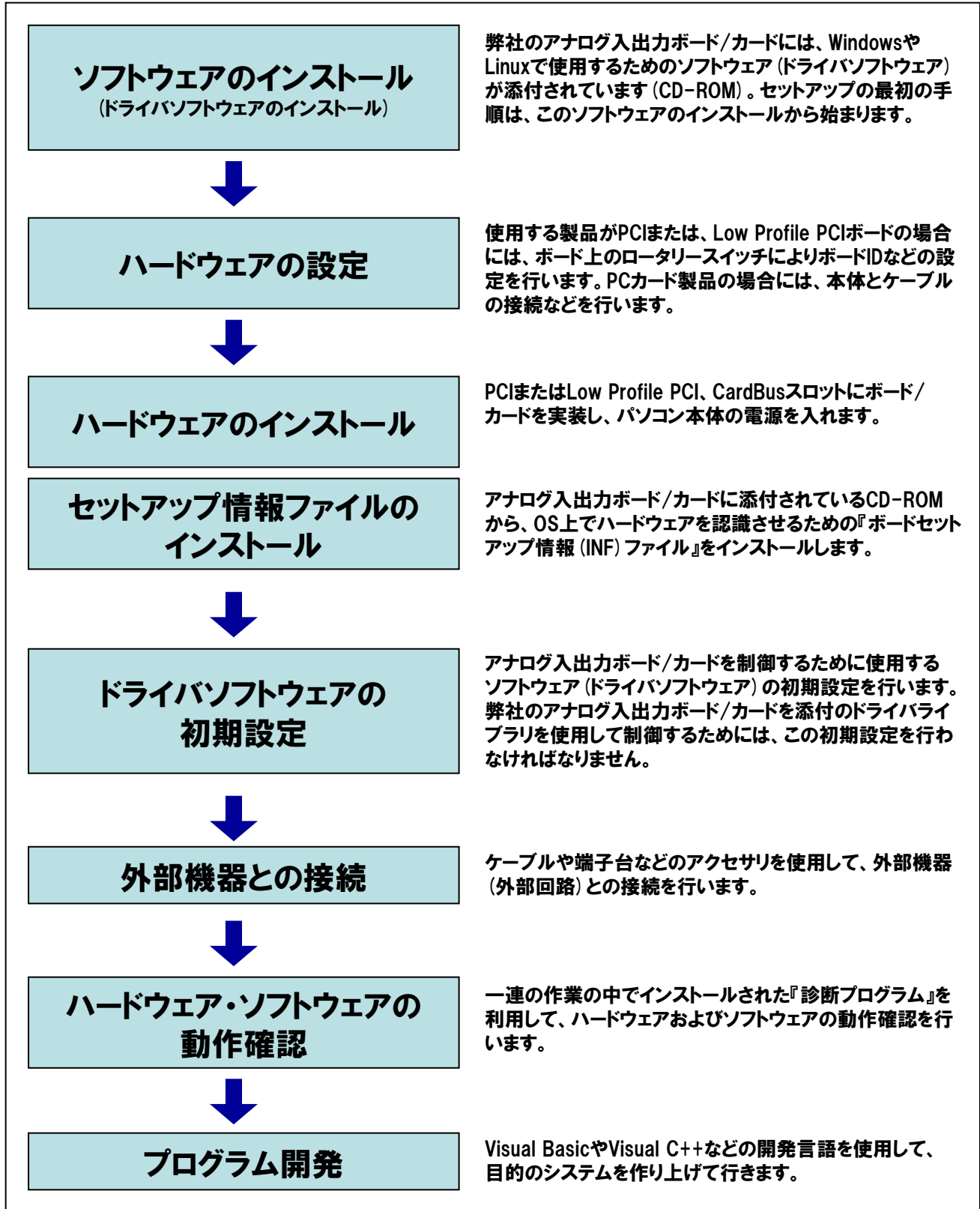
ハードウェア・ソフトウェアの セットアップ



第3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始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の場合を例にとりて解説を行います。

3-1. 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)

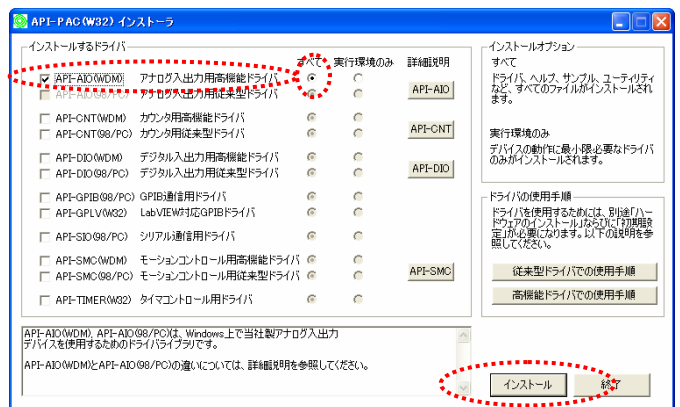


3-2.手順 ①:ソフトウェア (ドライバソフトウェア) のインストール

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います。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 2005
- インターフェイス : CardBus対応非絶縁型低価格高精度アナログ入出力カード 【型式:ADA16-8/2 (CB) L】
- ケーブル : 68ピン→50ピン変換シールドケーブル 【型式:ADC-68M/50M】
- アクセサリ : BNC端子台 【型式:ATP-8L】
- ソフトウェア : Windows版高機能アナログ入出力ドライバ (製品添付) 【型式:API-AIO (WDM)】

- 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 『API-PAC (W32) DRIVER LIBRARY』が自動的に表示されます。
- ③ 『API-PAC (W32) DRIVER LIBRARY』の『実行環境または開発環境のインストール』ボタンをクリックします。
- ④ 『インストーラ』画面が表示されます。
- ⑤ 『高機能アナログ入出力ドライバ API-AIO (WDM)』を選択します。
- ⑥ 『すべて』を選択します。
- ⑦ 『インストール』ボタンをクリックします。



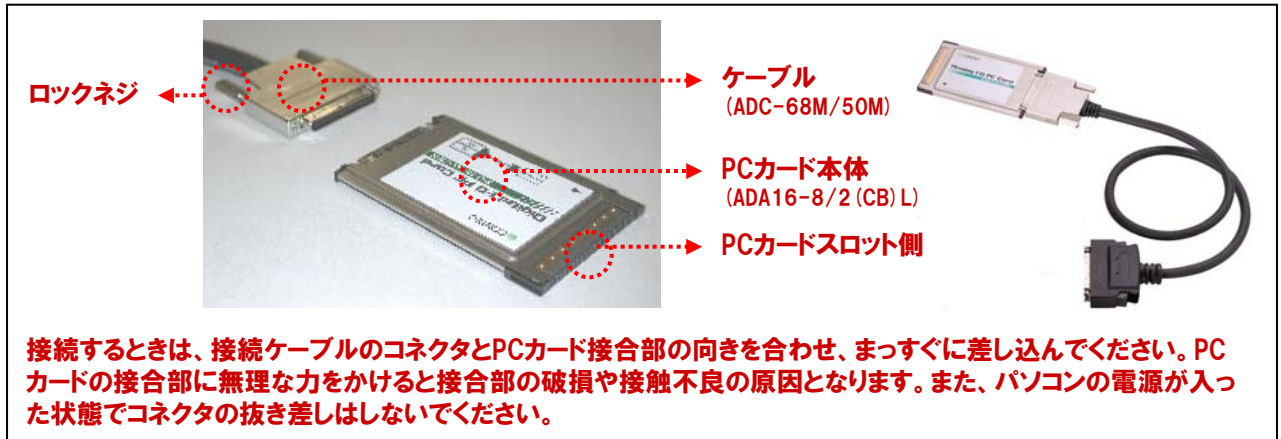
- ⑧ 画面の指示に従ってインストール作業を進めます。
- ⑨ インストール終了後、Readme (紹介) ファイルが表示されます。これでソフトウェアのインストールは完了です。



3-3.手順 ②:ハードウェアの設定

PCカード (ADA16-8/2 (CB) L) とケーブル (ADC-68M/50M) を接続し、パソコンに実装する手順を説明します。

- ① ケーブル (ADC-68M/50M) とPCカードを接続します。ケーブルのPCカード側コネクタとPCカード本体 (ADA16-8/2 (CB) L) 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。

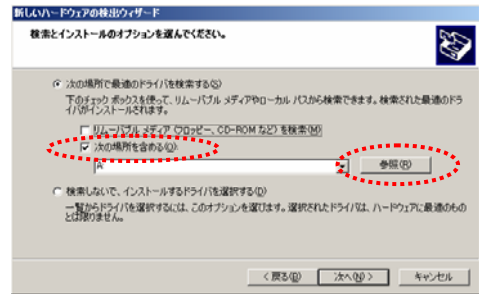
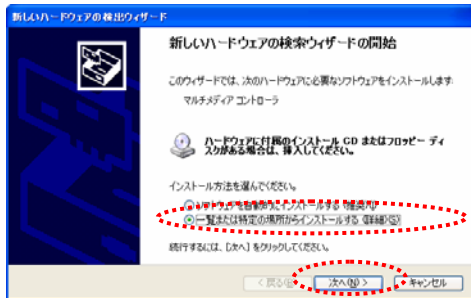
- ◎ PCカードを所定の方向および手順以外の方法で、挿入しないでください。
- ◎ 接続ケーブルまたはケーブルのコネクタをもって、PCカードを挿入しないでください。
- ◎ 接続ケーブルのコネクタを接続したまま、パソコンを移動しないでください。
- ◎ 接続ケーブルのコネクタを無理に引っ張るなどして、PCカードの接合部に力をかけないでください。
- ◎ 接続ケーブルのコネクタの上に、物を置かないでください。

- ③ これで、ハードウェアの設定は完了です。

3-4.手順 ③:ハードウェアのインストール

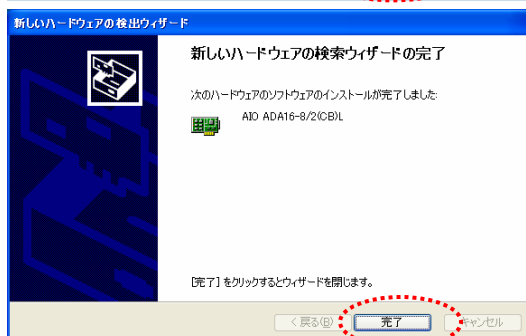
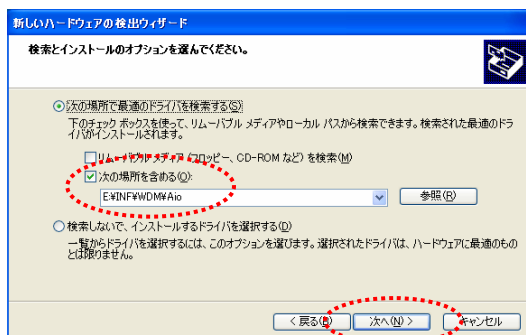
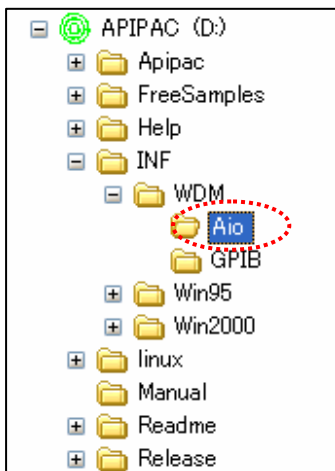
Windowsでは、PCカードが使用するI/Oアドレスと割り込みレベルをOSに認識させる必要があります。これをハードウェアのインストールと呼びます。複数枚のPCカードを使用する場合は、必ず1枚ずつ設定が完了してから次のPCカードをインストールしてください。

- ① パソコンの電源を投入します。
- 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。

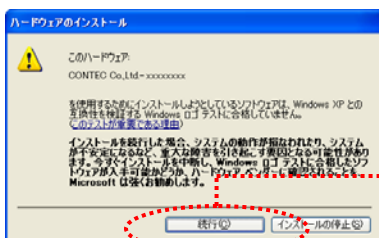


- ③ CD-ROMからセットアップ情報(INF)ファイルのあるフォルダを指定して、登録を行います。セットアップ情報(INF)ファイルは、添付CD-ROMの以下のフォルダにありますので、そのフォルダを指定して『OK』をクリックします。ADA16-8/2(CB)Lがパソコンに設定されますので、完了をクリックします。これで、インストールは完了です。

● ¥INF¥WDM¥Aio



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きる場合があります。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。

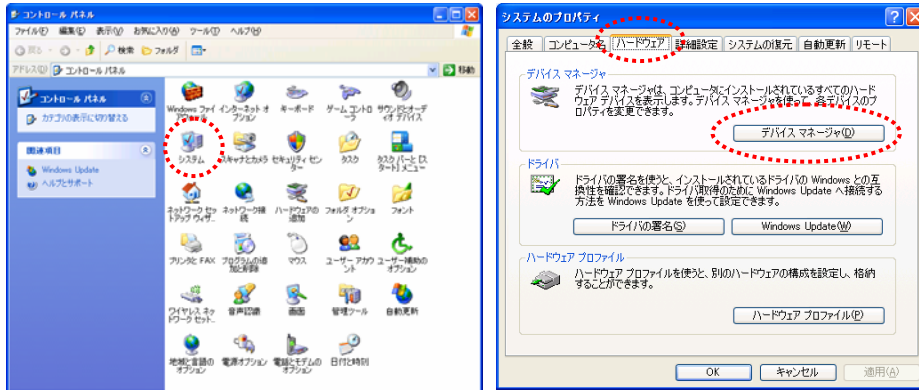


→ [続行] ボタンをクリックしてください

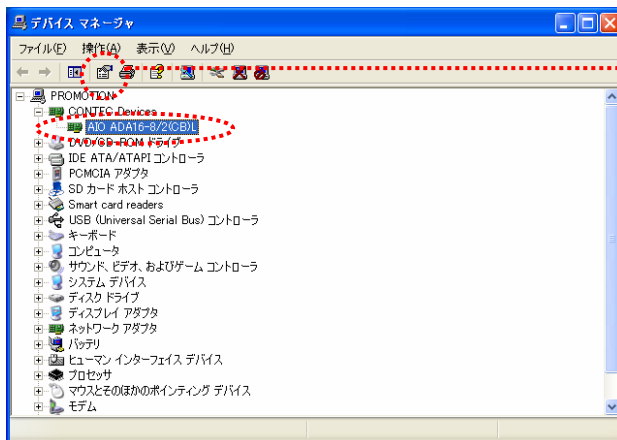
3-5.手順 ④:ドライバソフトウェアの初期設定

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。弊社のアナログ入出力ボード/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、アナログ入出力ボード/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 デバイス名の設定を行います。Windows XPを使用している場合、次の手順で『デバイスマネージャ』を起動します。
『スタート』 - 『コントロールパネル』 - 『システム』 - 『ハードウェア』 - 『デバイスマネージャ』

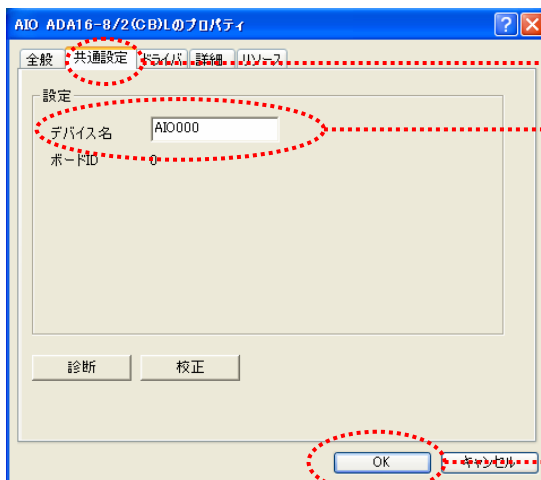


- ② インストールしたハードウェアは、『CONTEC Devicesツリー』の下に登録されています。デバイストリーを開き、ADA16-8/2 (CB)Lを選択して反転表示させてください。[プロパティ] をクリックします。



→ [プロパティ] ボタンをクリックしてください。

- ③ デバイスのプロパティページが表示されます。『共通設定タブ』でデバイス名を入力して [OK] をクリックしてください。ここで設定するデバイス名は、プログラミング時に必要です。最初に表示されているデバイス名は初期値です。変更しても構いませんが、デバイス名は、複数のデバイス間で重複しないように設定してください。本書では、初期値を使用します。これでソフトウェアの初期設定は完了です。



→ [共通設定] タブをクリックしてください。

→ [デバイス名] を設定します。
本書では、初期値『AIO000』を使用します。

→ デバイス名設定後には、必ず『OKボタン』をクリックしてください。

3-6.手順 ⑤:外部機器との接続

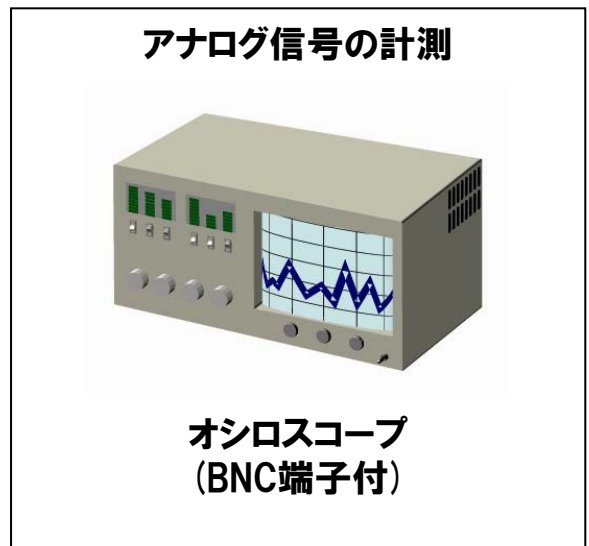
本書においては、センサまたはアクチュエータの代わりに外部機器として、ファンクションジェネレータ(波形発生器)および、オシロスコープを使用します。実システムの場合には、ボード/カードのインターフェイスコネクタと外部機器(センサなどを使用した外部回路)とをケーブルまたは端子台を利用して適切に接続してください。

3-6-1.外部機器との接続形態(接続方法)

本カード(ADA16-8/2(CB)L)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線で接続する方法のいずれかとなります。



3-6-2.本書にて使用する外部機器

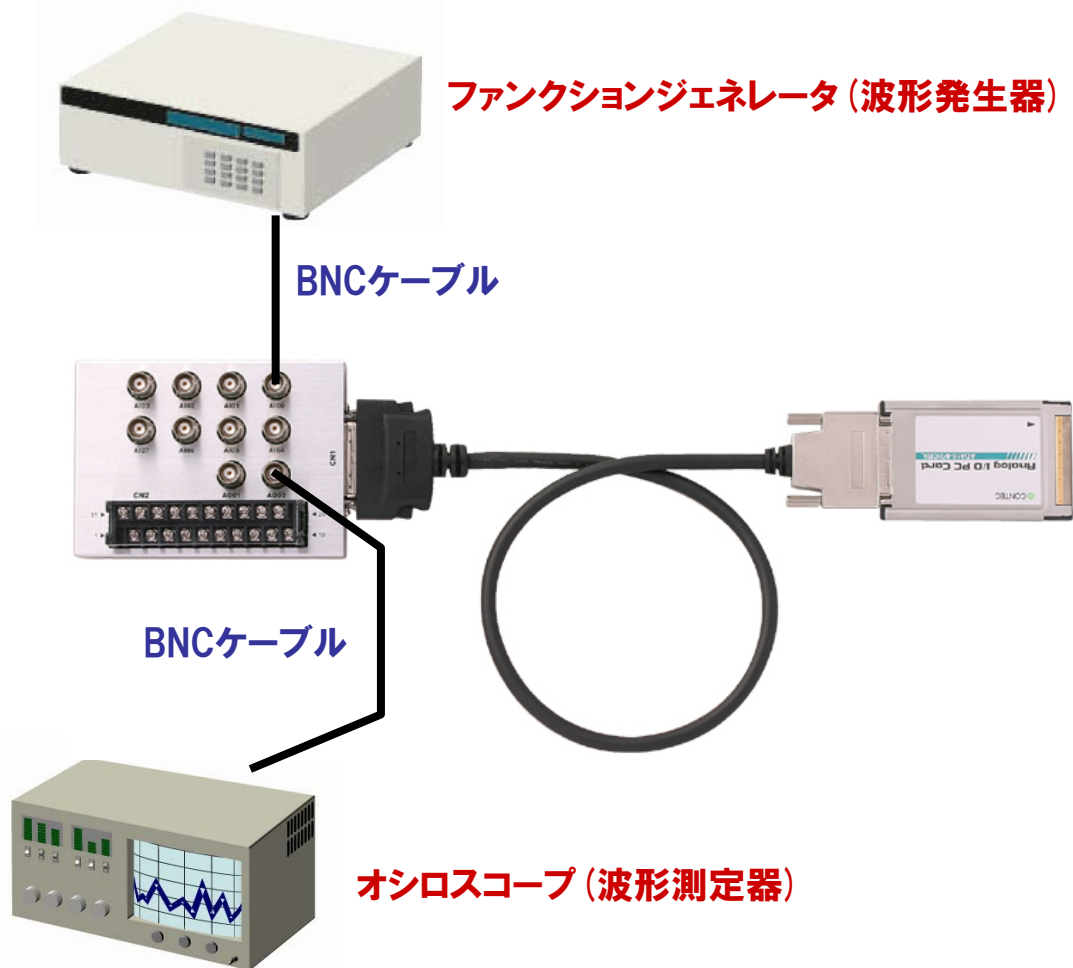


※ 外部機器は、お客様にてご用意ください。

3-6-3. 実習環境の構築

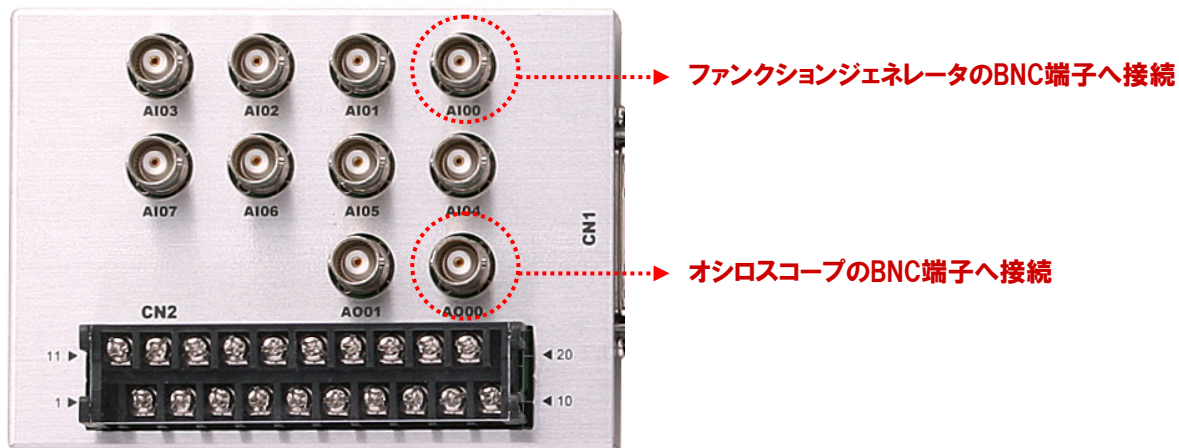
ボードの動作確認および、第4、第5章のアナログ入出力プログラミングで使用する環境を構築するため、ケーブルとBNC端子台を接続します。その後、端子台のBNCコネクタと外部機器のBNC端子をBNCケーブルで接続します。

接続イメージ図



BNC端子台: ATP-8Lの端子図

本書では、アナログ入力の『0ch (端子名: AI00)』とアナログ出力の『0ch (端子名: AO00)』を使用します。



3-6-4.参考資料:圧着用端子台を使用して外部機器と接続する方法

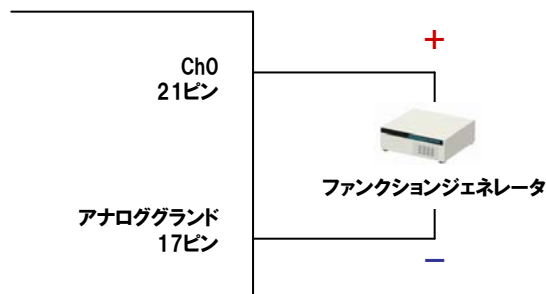
8ピン→50ピン変換シールドケーブル(型式:ADC-68M/50M)と圧着用端子台(型式:EPD-50A)を使用して外部機器と接続する場合には、コネクタの信号配置図(端子台のピン番号と1対1)を参照して行います。

ADC-68M/50Mを使用した場合の信号配置

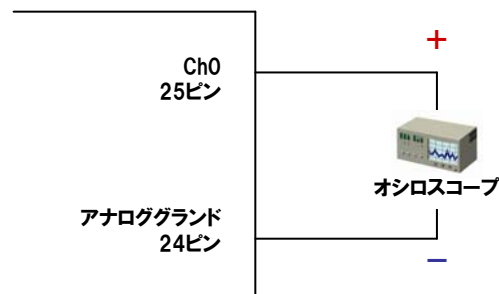
Analog Output 00	AO 00	25	50	- N.C.	Non Connect
Analog Ground (for AO)	AGND	24	49	- AGND	Analog Ground (for AO)
Analog Output 01	AO 01	23	48	- N.C.	Non Connect
Analog Ground (for AO)	AGND	22	47	- AGND	Analog Ground (for AO)
Analog Input 00	AI 00	21	46	- AI 04	Analog Input 04
Non Connect	N.C.	20	45	- N.C.	Non Connect
Analog Input 01	AI 01	19	44	- AI 05	Analog Input 05
Non Connect	N.C.	18	43	- N.C.	Non Connect
Analog Ground (for AI)	AGND	17	42	- AGND	Analog Ground (for AI)
Analog Ground (for AI)	AGND	16	41	- AGND	Analog Ground (for AI)
Analog Input 02	AI 02	15	40	- AI 06	Analog Input 06
Non Connect	N.C.	14	39	- N.C.	Non Connect
Analog Input 03	AI 03	13	38	- AI 07	Analog Input 07
Non Connect	N.C.	12	37	- N.C.	Non Connect
AI External Start Trigger Input	AI START	11	36	- AO START	AO External Start Trigger Input
AI External Stop Trigger Input	AI STOP	10	35	- AO STOP	AO External Stop Trigger Input
AI External Sampling Clock Input	AI EXCLK	9	34	- AO EXCLK	AO External Sampling Clock Input
Digital Ground	DGND	8	33	- DGND	Digital Ground
Digital Input 00	DI 00	7	32	- DO 00	Digital Output 00
Digital Input 01	DI 01	6	31	- DO 01	Digital Output 01
Digital Input 02	DI 02	5	30	- DO 02	Digital Output 02
Digital Input 03	DI 03	4	29	- DO 03	Digital Output 03
Digital Ground	DGND	3	28	- DGND	Digital Ground
Counter Gate Control Input	CNT GATE	2	27	- CNT UPCLK	Counter UP Clock Input
Counter Output	CNT OUT	1	26	- Reseved	Reserved

Analog Input00 to Analog Input07	アナログ入力信号です。番号はチャンネル番号に対応します。
Analog Output00 to Analog Output01	アナログ出力信号です。番号はチャンネル番号に対応します。
Analog Ground	アナログ入出力信号に共通のアナロググランドです。
AI External Start Trigger Input	アナログ入力用サンプリング開始条件の外部トリガ入力信号です。
AI External Stop Trigger Input	アナログ入力用サンプリング停止条件の外部トリガ入力信号です。
AI External Sampling Clock Input	アナログ入力用外部サンプリングクロック入力信号です
AO External Start Trigger Input	アナログ出力用サンプリング開始条件の外部トリガ入力信号です。
AO External Stop Trigger Input	アナログ出力用サンプリング停止条件の外部トリガ入力信号です。
AO External Sampling Clock Input	アナログ出力用外部サンプリングクロック入力信号です
Digital Input00 to Digital Input03	デジタル入力信号です。
Digital Output00 to Digital Output03	デジタル出力信号です。
Counter Gate Control Input	カウンタのゲート制御入力信号です。
Counter Up Clock Input	カウンタのアップクロック入力信号です。
Counter Output	カウンタの出力信号です。
Digital Ground	デジタル入出力信号、外部トリガ入力信号、外部サンプリングクロック入力信号、カウンタ入出力信号に共通のデジタルグランドです。
Reserved	このピンは予約です。
N.C.	このピンはどこにも接続されていません。

アナログ入力の『0ch』を使用する場合の配線図



アナログ出力の『0ch』を使用する場合の配線図

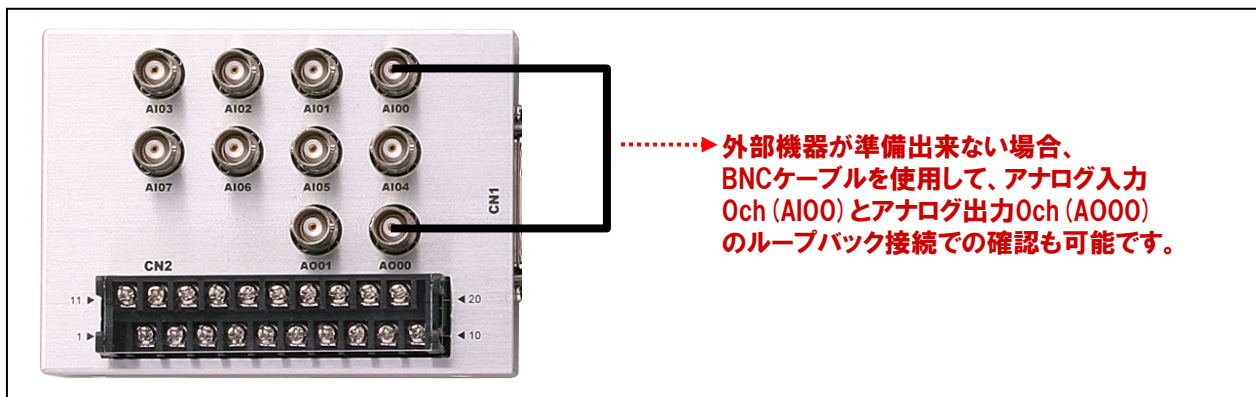


3-7.手順 ⑥:ハードウェア・ソフトウェアの動作確認

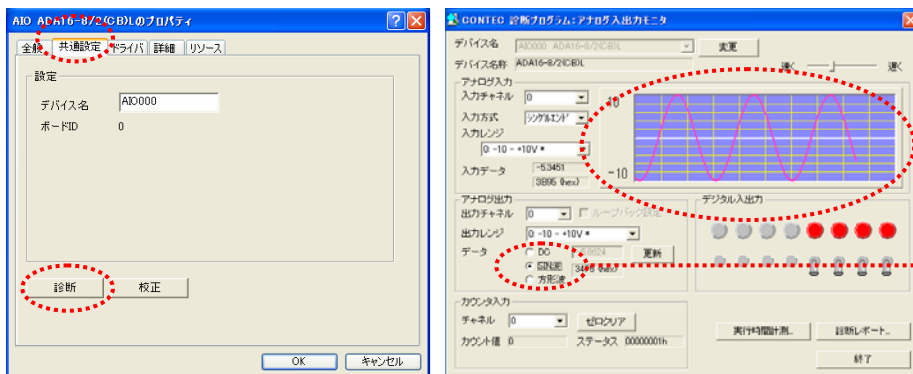
診断プログラムを使用して、ハードウェアやドライバソフトウェアが正常動作することを確認します。これでセットアップが正しくできたことを確認できます。診断プログラムは、ボードとドライバソフトウェアの状態を診断するプログラムです。実際に外部機器を接続したときの簡易動作確認として使用することもできます。『診断レポート』機能を使用すれば、ドライバ設定、ボード存在有無、I/O状況、割り込み状況がレポートとして作成されます。

3-7-1.確認方法

【3-6-3.】の構成で、ADA16-8/2 (CB) Lとドライバソフトウェアの動作確認を行います。この際使用する、ファンクションジェネレータや、オシロスコープの操作方法は、それら機器の説明書で確認してください。それらの機器が用意できない場合、アナログ入力0ch (AI00) とアナログ出力0ch (A000) をBNCケーブルで接続した (ループバック) 診断プログラムでの動作確認も可能です。

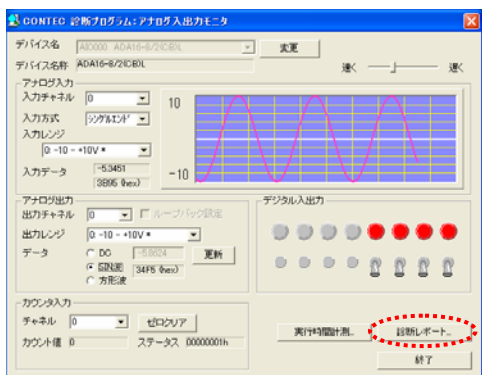


- ① 【3-5.手順 ④:ドライバソフトウェアの初期設定】の手順で、『デバイスのプロパティ』を表示させてください。『共通設定タブ』の『診断ボタン』をクリックし、診断プログラムを立ち上げると診断が開始されます。



アナログ出力では、SIN波・方形波、DC一定出力が行えます。ループバックしている場合は、アナログ入力のグラフ画面で視覚的に動作を確認することができます。

- ② 診断プログラムのメイン画面の『診断レポート』ボタンをクリックすると動作確認した結果のレポートが表示されます。診断レポートではOSのバージョン、デバイス情報、ファイル情報、初期化、割り込み、各チャンネルの入出力状態などの各種情報が確認できます。

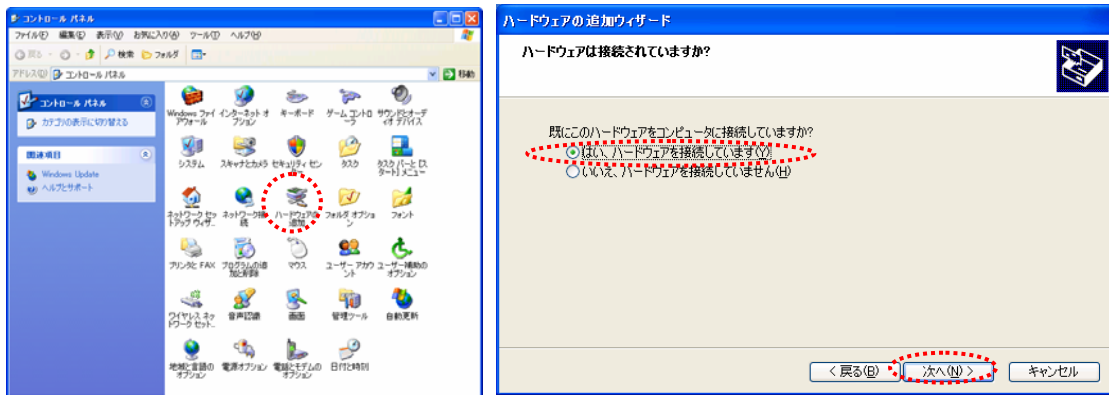


3-8. DEMO DEVICE (仮想ハードウェア) の登録・設定方法

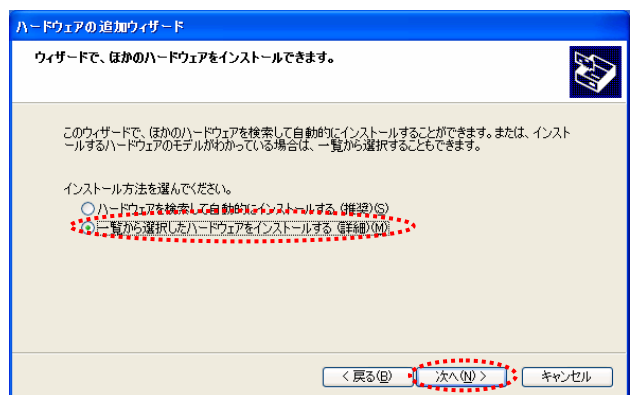
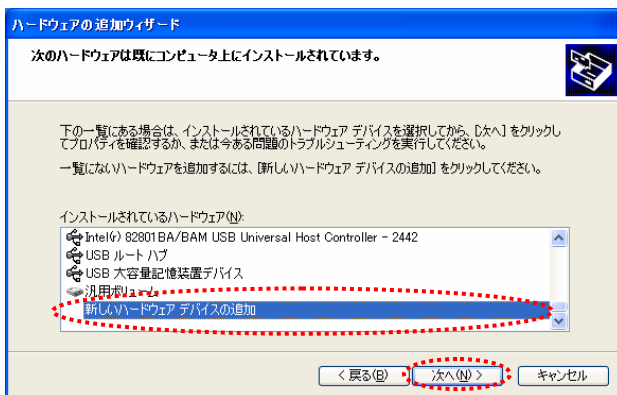
実際のハードウェア (ボードまたはカード) を用意できない場合、ドライバソフトウェアに含まれているDEMO DEVICE (仮想ハードウェア) を使用することによって、プログラムの動作確認を行うことができます。ここでは、デモボードの登録方法を解説します。なお、実際のハードウェアを使用する際には、本項の設定は必要ありません。

3-8-1. デモボードの登録方法

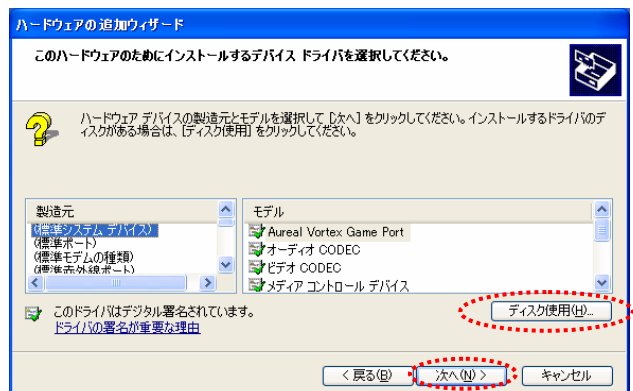
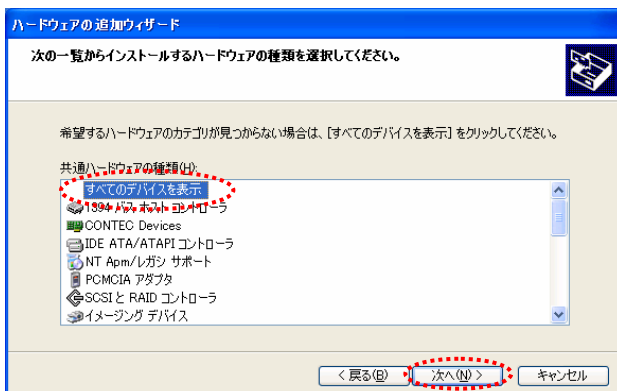
- ① Windows XPを使用している場合、次の手順で『ハードウェアの追加ウィザード』を起動します。
『スタート』 - 『コントロールパネル』 - 『ハードウェアの追加』
次の画面で [はい、ハードウェアを接続しています] をクリックします。



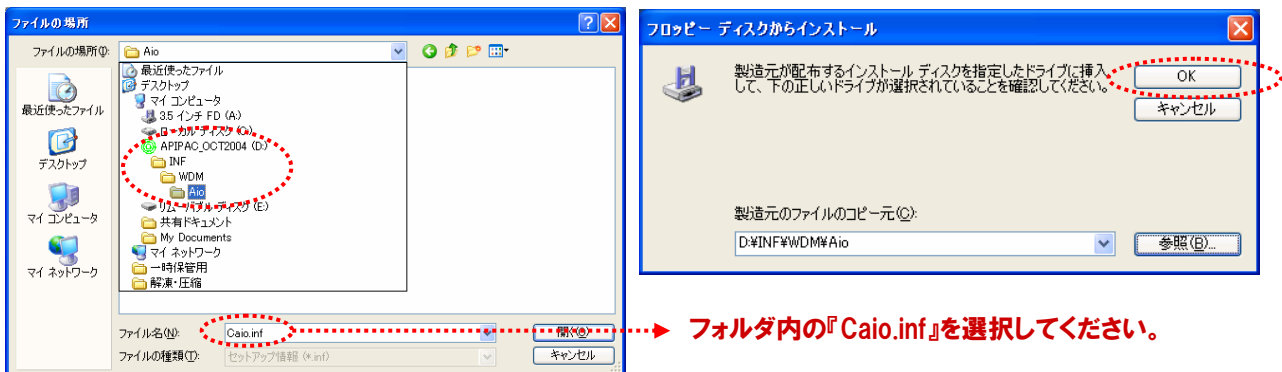
- ② インストールされているハードウェアから、[新しいハードウェア デバイスの追加] を選択します。
- ③ [一覧から選択したハードウェアをインストールする] を選択します。



- ④ インストールするハードウェアの種類から、[すべてのデバイスを表示] を選択します。
- ⑤ 以下の画面では、[ディスク使用] をクリックします。



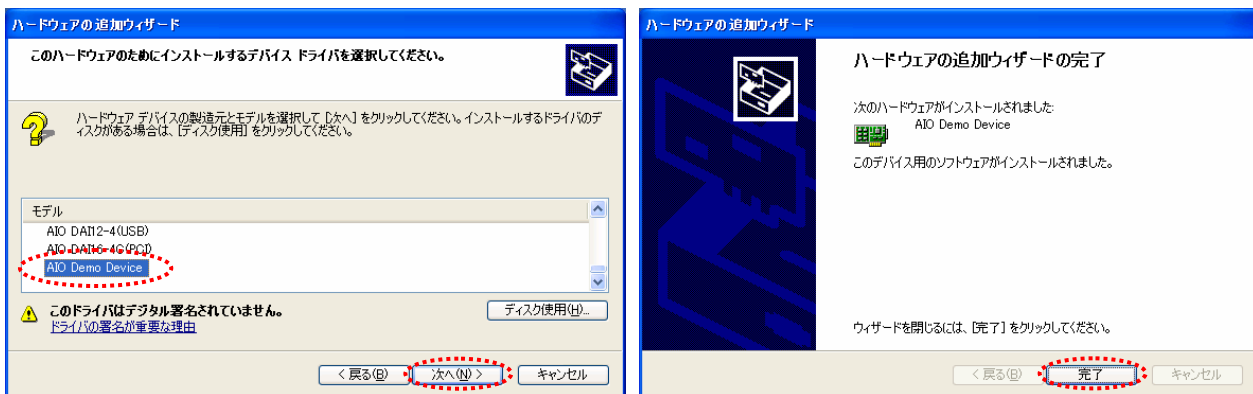
⑥ CD-ROMドライブ ¥ INF ¥ WDM ¥ AIOを指定します。次の画面では、[OK] ボタンをクリックします。



フォルダ内の『Caio.inf』を選択してください。

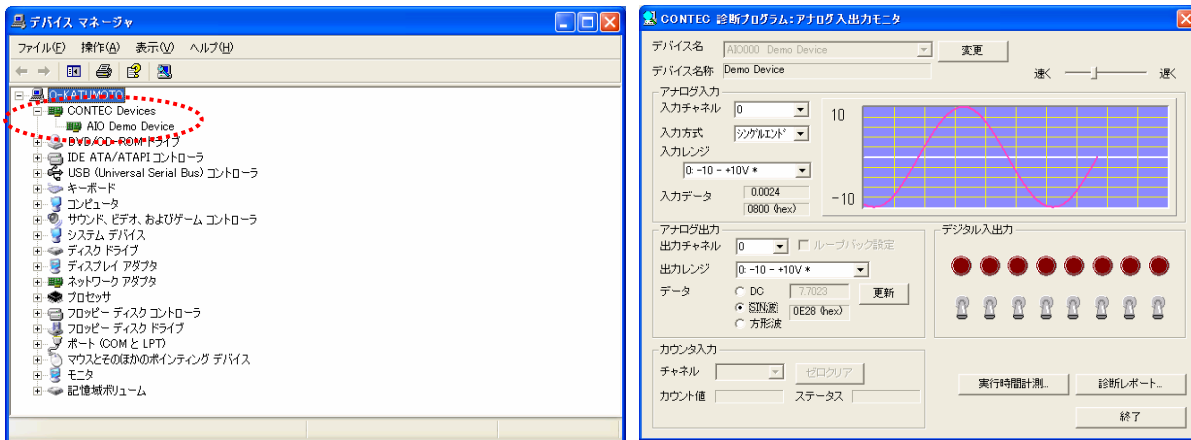
⑦ ハードウェアの一覧から、[AIO Demo Device] を選択して [次へ] をクリックします。

⑧ 最後に以下の画面が表示されればインストールは終了です。



⑨ [デバイスマネージャ] 上で、[CONTEC Devices] の下に [AIO Demo Device] が確認できます。

[AIO Demo Device] の動作確認も診断プログラムで行えます。



3-8-2. 補足

Demo Deviceは、ドライバ中で作成された仮想的なデバイスです。実際のデバイスがない状態でもプログラミングを行い、動作させることが可能です。Demo Deviceのアナログ入力、以下のデータを擬似的に作成しています。

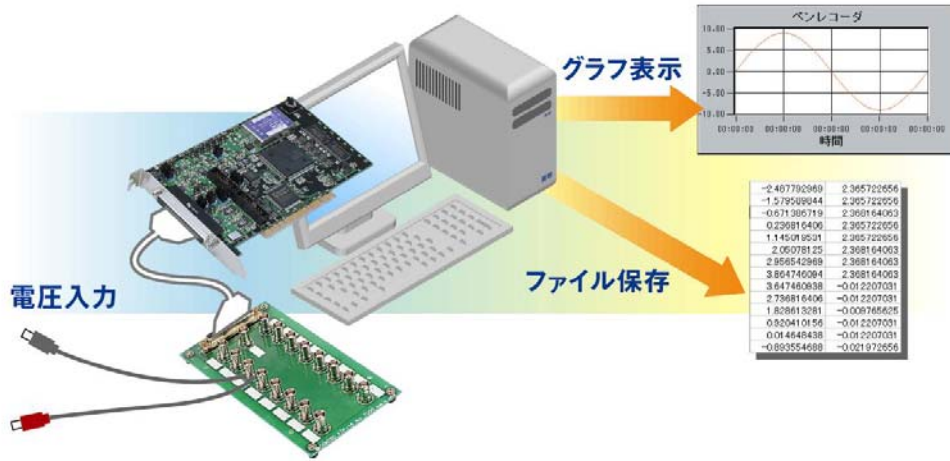
- 0, 4, 8 …チャンネル:サイン波形
- 1, 5, 9 …チャンネル:コサイン波形
- 2, 6, 10…チャンネル:のこぎり波形
- 3, 7, 11…チャンネル:方形波

Demo Deviceのデジタル入出力は、同じチャンネルの入力端子と出力端子が短絡されている状態になります。例えば、0チャンネルのデジタル出力をONすると、0チャンネルのデジタル入力がONになります。Demo Deviceの仕様などの詳細情報は、API-AIO (WDM) のヘルプファイルを参照してください。

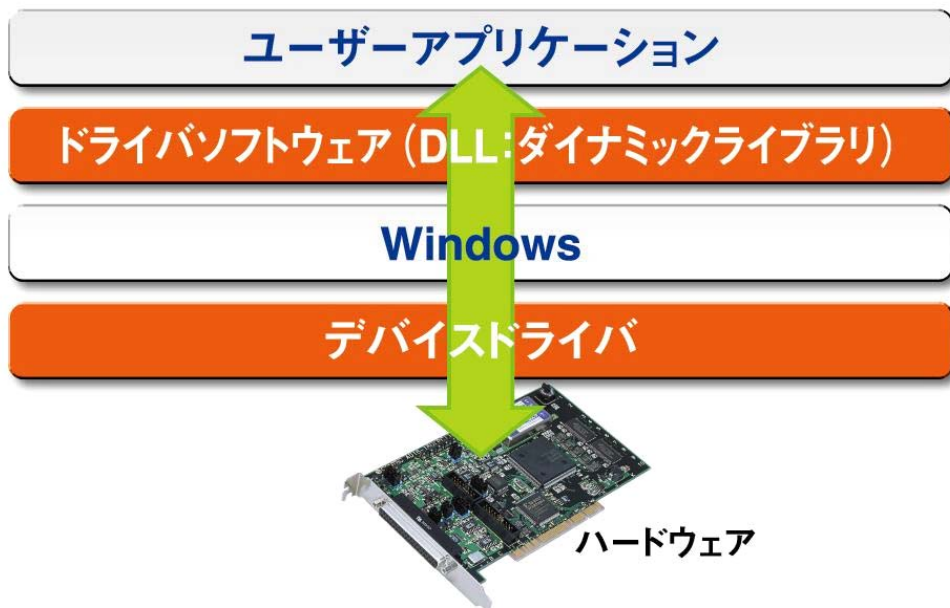
3-9. 参考資料 (ドライバソフトウェアの必要性)

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。



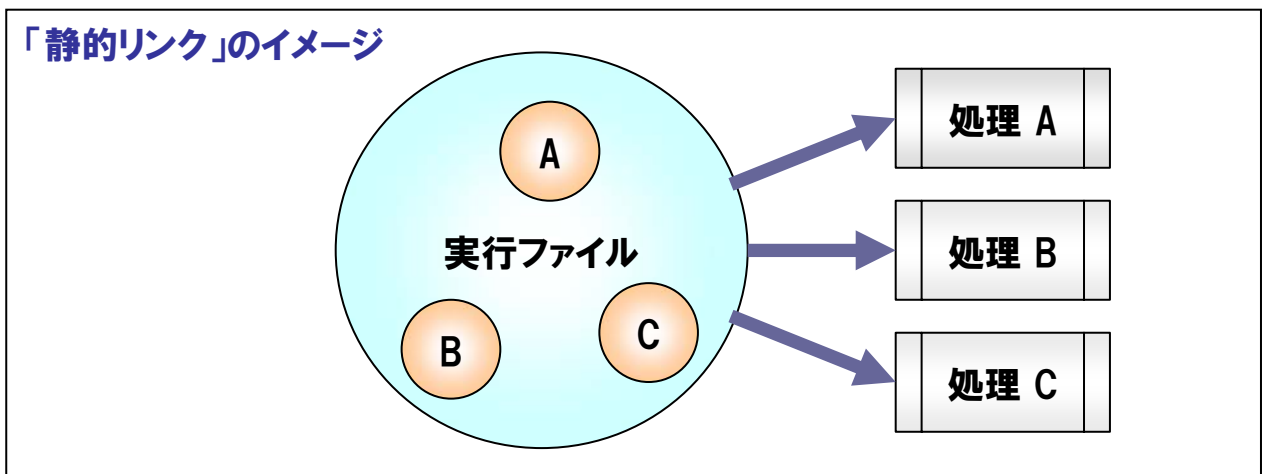
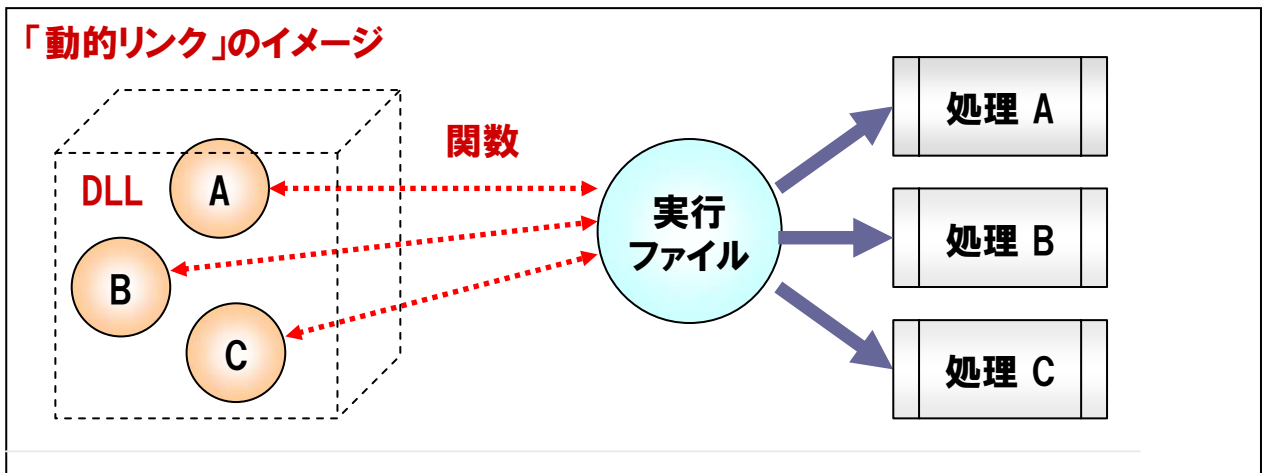
現在、さまざまな機能を持った各種ハードウェアが多数のメーカーから販売されていますが、仮に高機能ハードウェアであっても、それを操作するドライバソフトウェア (関数など) が使いやすく設計されていなければ、結局は、使いにくい物になってしまうということです。すなわちWindows環境では、ハードウェアが持っている機能はもとより、どれだけ使い易いソフトウェアが提供されているかが、製品選定の重要な要素となっているのです。

3-10. 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。

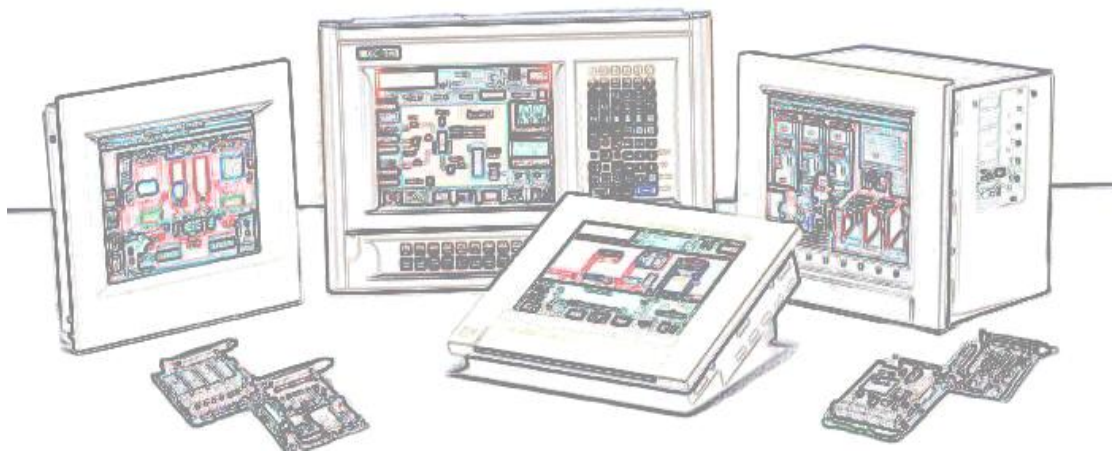
3-10-1. Windows環境における実行ファイル (.EXEファイル)

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4章

Visual Basic2005による アナログ入出力プログラミング



第4章 Visual Basic2005によるデジタル入出力プログラミング

第3章までの解説と準備で、アナログ入出力のプログラミング環境が整いました。本章では、Visual Basic2005を使用して、外部からのアナログ信号入力とパソコンから外部機器へのアナログ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 Basicに関して簡単な説明をしておきましょう。

4-1. Visual Basicでのプログラム開発

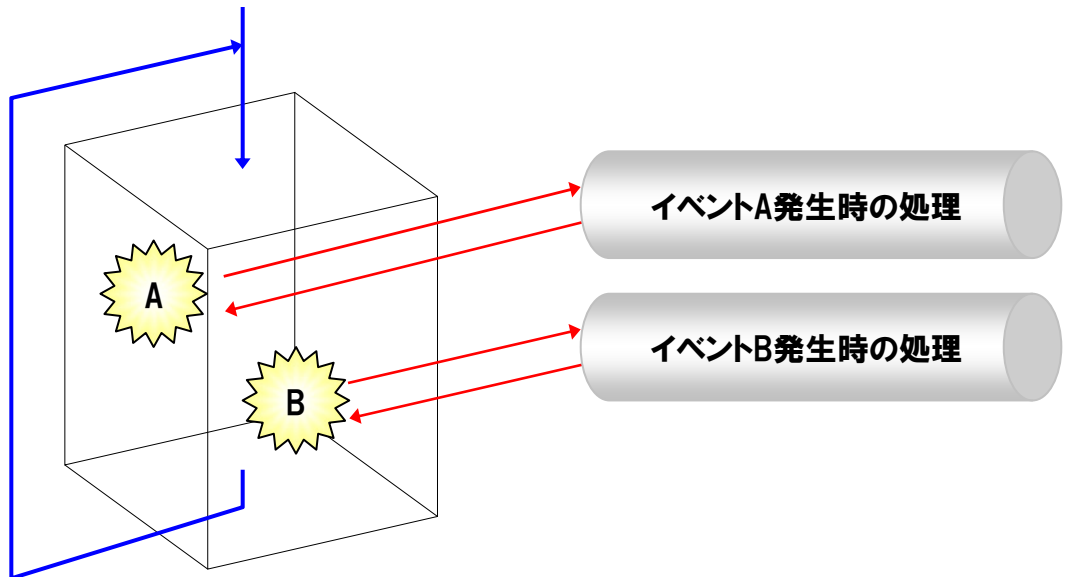
Visual Basicは、GUI (Graphical User Interface) アプリケーションを簡単に作成するため、従来の『手続き型プログラミング』ではなく、『イベント駆動型 (イベントドリブン) プログラミング』という手法を採用しています。Visual Basicは、米国Microsoft社によって開発されたWindowsアプリケーションを簡単に作成できるプログラミング言語です。

4-1-1. 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。

4-1-2. Visual Basic2005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『.vb』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。

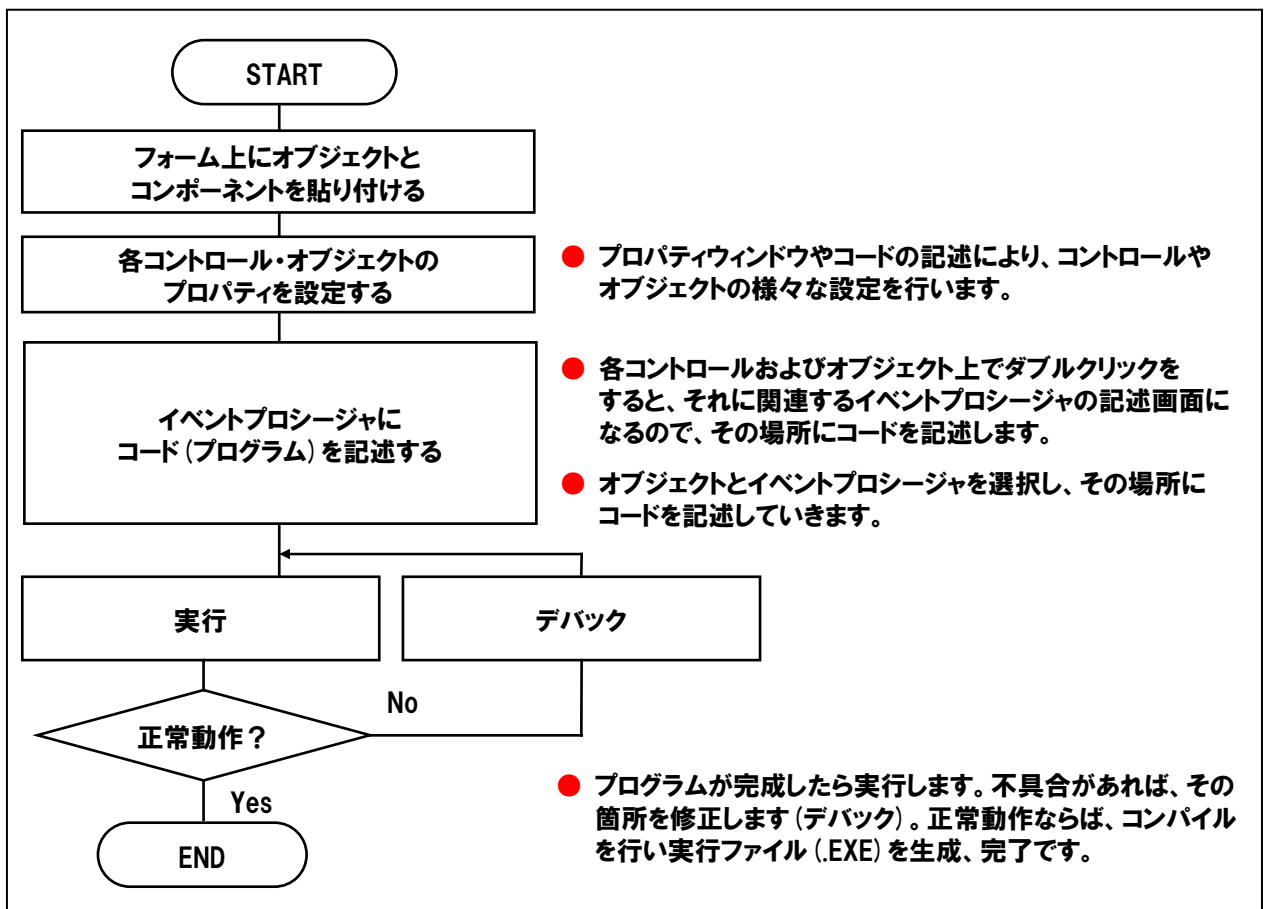
⑤イベントプロシージャ

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』『ウィンドウプロシージャ』『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。

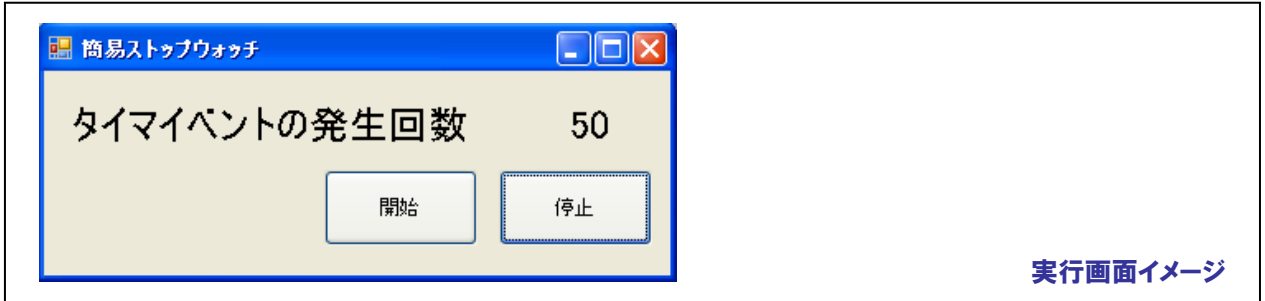
5-1-3. Visual Basicのプログラム作成手順



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。

4-2.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。



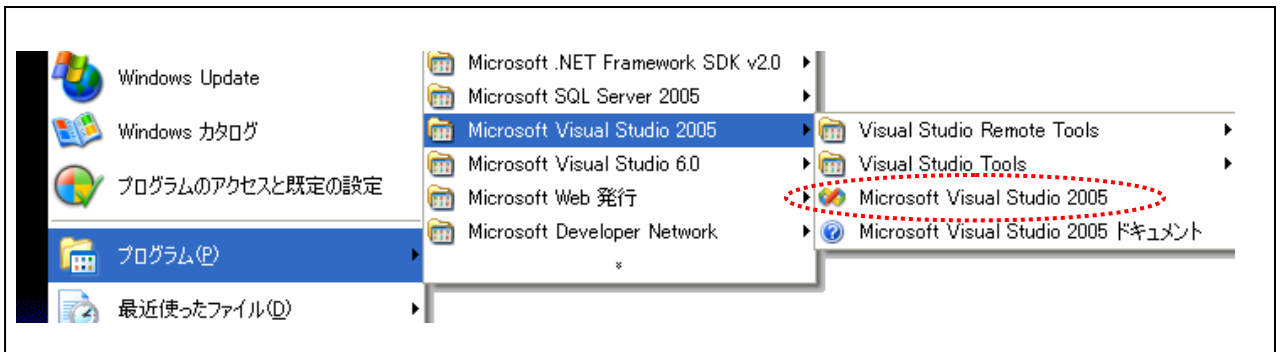
4-2-1.プログラム フローチャート



4-2-2.プログラム作成手順① (Visual Basicの起動)

まず最初にVisual Studio2005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。

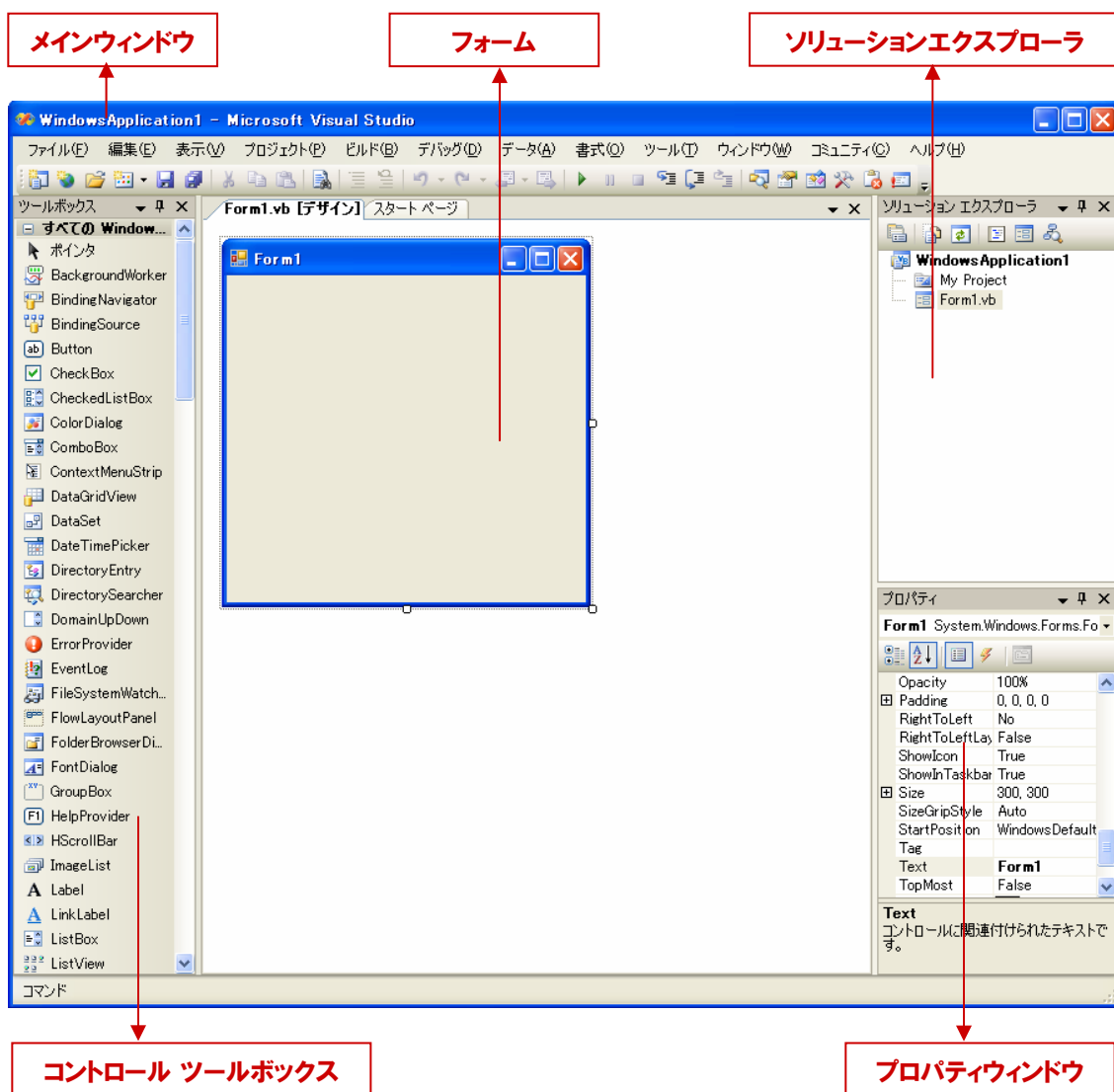
『スタート』 - 『プログラム』 - 『Microsoft Visual Studio 2005』 - 『Microsoft Visual Studio 2005』を選択します。



メニューの『ファイル』 - 『新しいプロジェクト』を選択します。『新しいプロジェクト』ダイアログが表示されますので、『プロジェクトの種類』から『Visual Basic』 - 『Windows』を選択し、『テンプレート』から『Windows アプリケーション』を選択して、『OK』ボタンをクリックしてください。



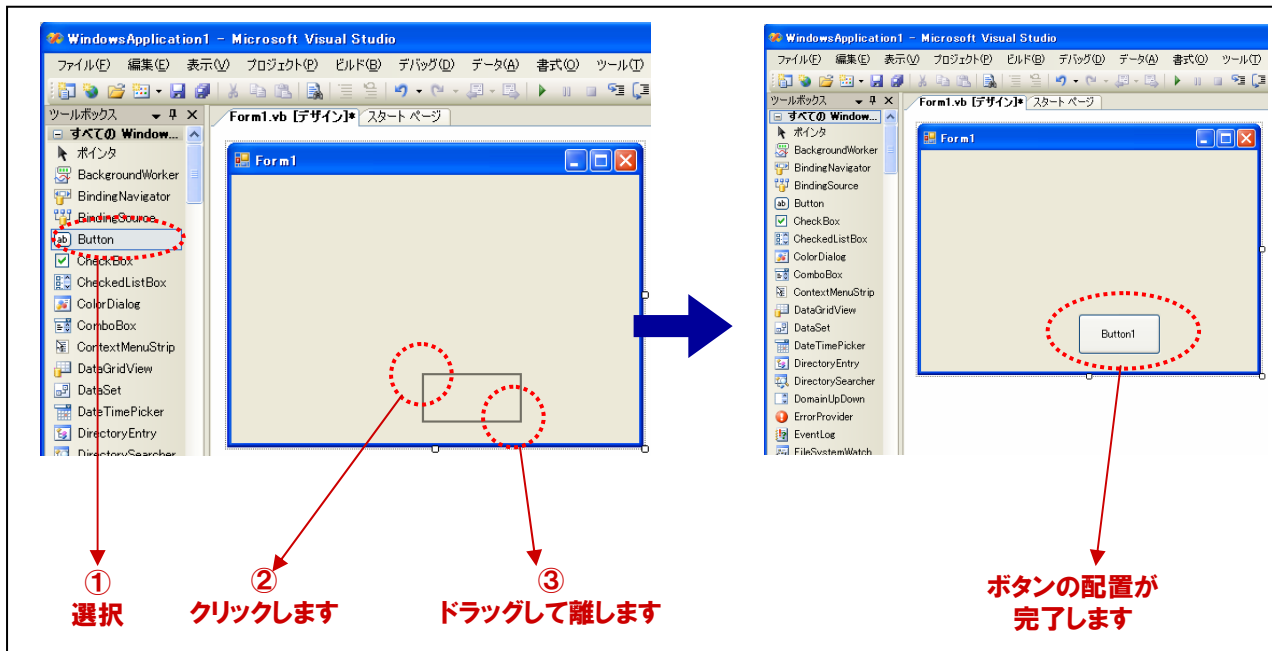
Visual Basic2005の画面構成



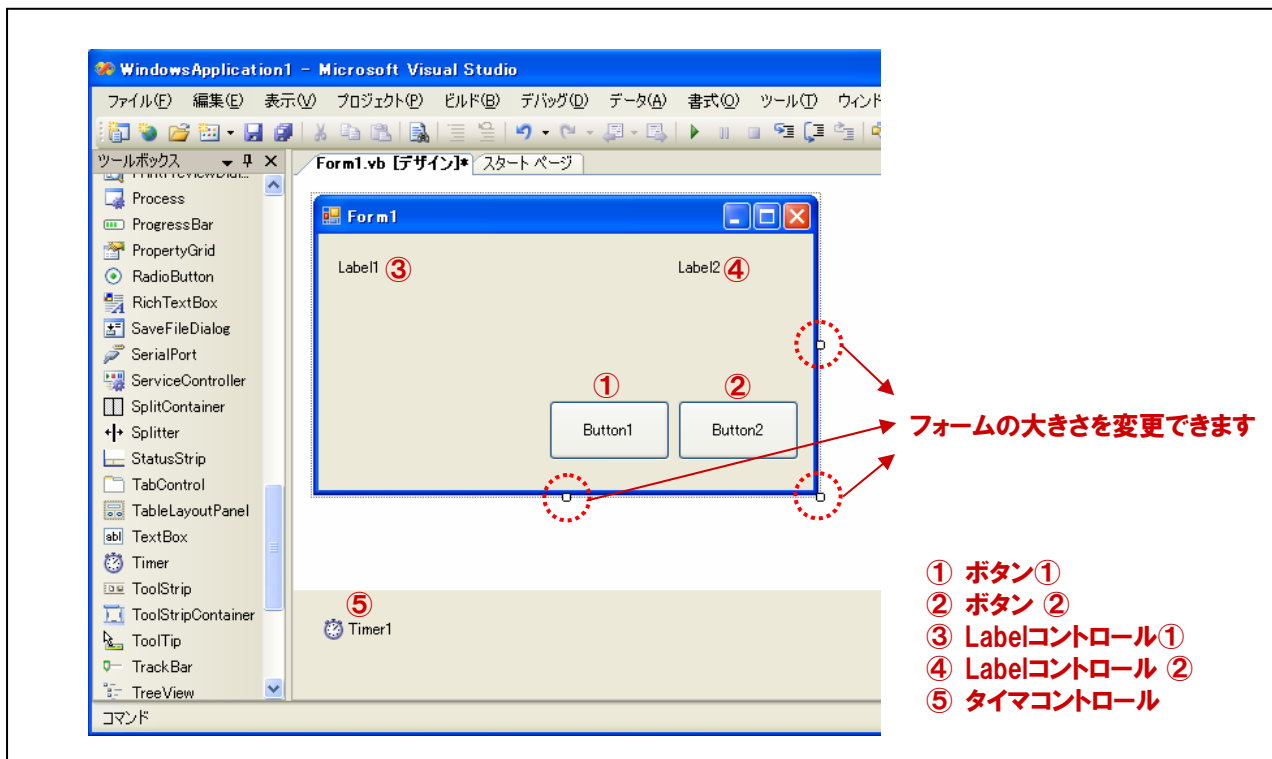
4-2-3.プログラム作成手順② (コントロールの配置)

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『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『ボタン』を選択します。
- 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、『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、『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。



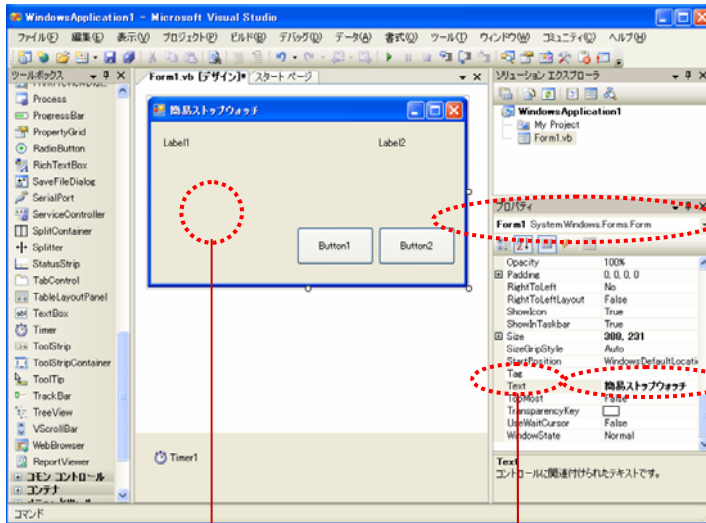
- ⑤ 同様の手順にて、2つ目の『ボタン』と2つの『Labelコントロール』、『タイマコントロール』をフォーム上に配置します。



4-2-4.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『Text』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。

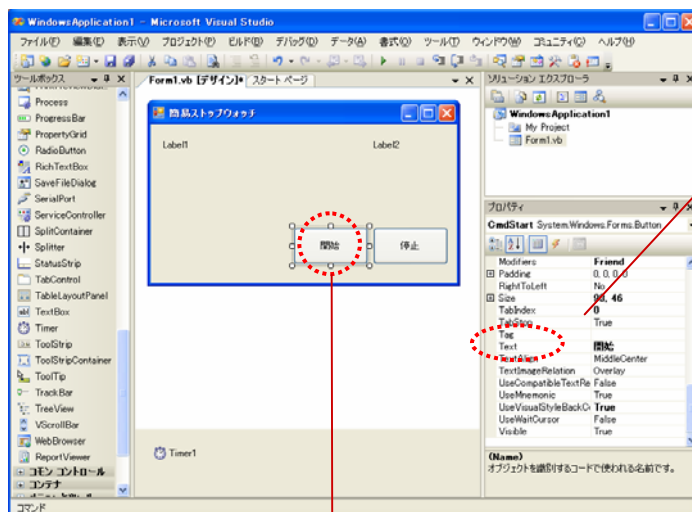
フォームをクリックし、
フォームを選択状態にします。

『Text』を選択します。

4-2-5.プログラム作成手順④ (ボタンコントロールのプロパティ設定)

ボタンの『オブジェクト名』と『タイトル』を変更します。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『(Name)』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、『Text』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。

同様にボタン②は、Nameに『cmdStop』、Textプロパティに『停止』と入力します。



ボタン①のプロパティ設定

○Name : cmdStart
○Text : 開始

ボタン②のプロパティ設定

○Name : cmdStop
○Text : 停止

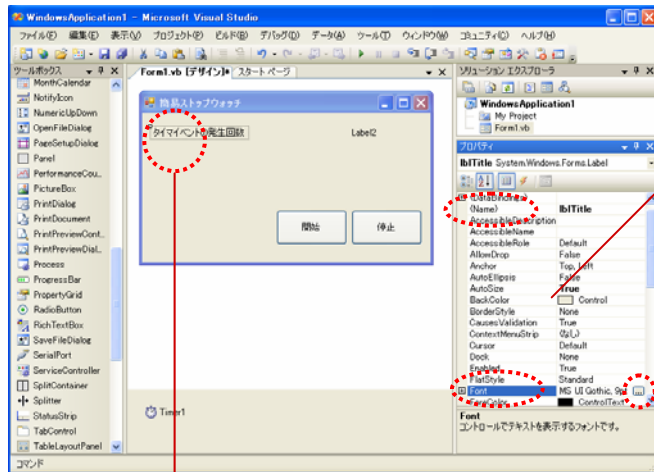
コマンドボタンをクリックし、
選択状態にします。

4-2-6.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』、『タイトル』、『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『(Name)』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、『Text』プロパティを選択して、設定ボックスに『タイマ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、Nameに『lblData』、Textプロパティに『0』と入力します。



Labelコントロール①のプロパティ設定

- ◎Name : lblTitle
- ◎Text : タイマ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

- ◎Name : lblData
- ◎Text : 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。



Labelコントロールをクリックし、選択状態にします。

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なるようにしています。

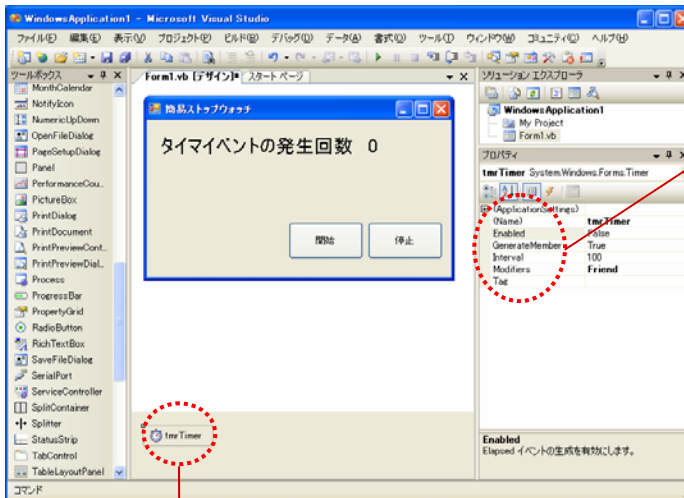
■主なオブジェクト名とプリフィックスとの関係

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

4-2-7.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『(Name)』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

◎Name :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。

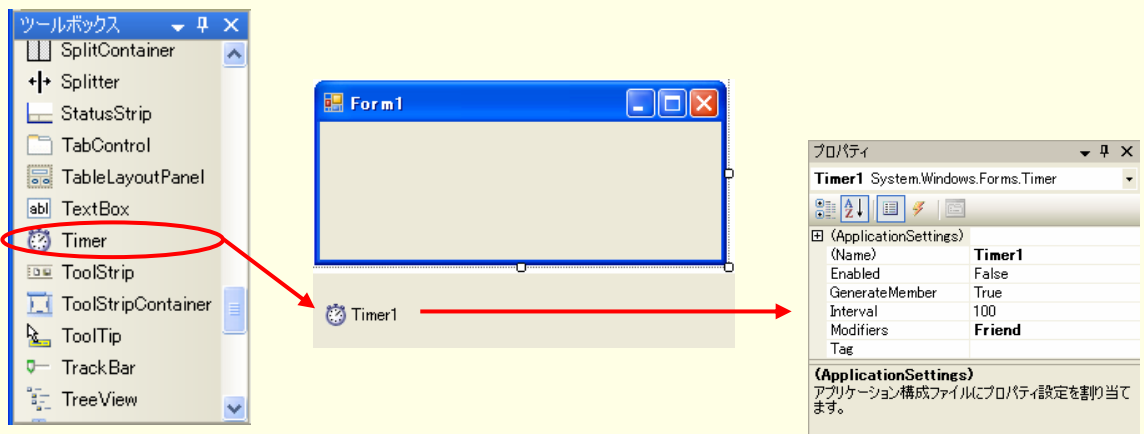
TOPICS

『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。

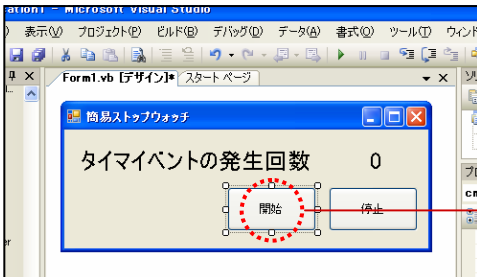
『Interval』プロパティ: タイマイベントが発生する間隔をmsec単位で設定します。『0~65535』の範囲で設定します。本プログラムでは、プログラム概要ののっとり、『100msec (0.1秒)』と設定しています。



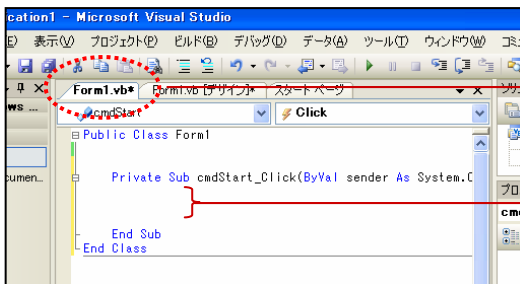
4-2-8.プログラム作成手順⑦(タイマ開始処理の記述)

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(Name: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(Name: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。

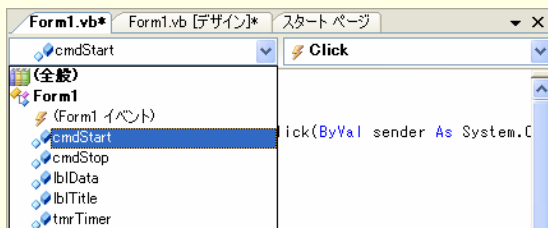
③ Private Sub cmdStart_Click() から、End Subまでの間に、何を行いたいかをコードで記述します。

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『何を』、『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。

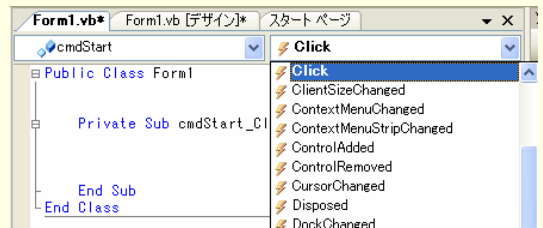
```
tmrTimer.Enabled = True
```

『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。例えば言えば『何を(が)』を選択する場所です。

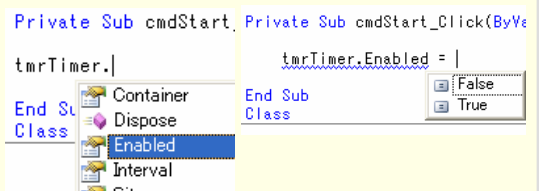


プロシージャボックスには、選択したオブジェクトが持っているプロシージャを選択することができます。例えば言えば『何した時に』を選択する場所です。

TOPICS

『Visual Basicのインテリセンス機能』

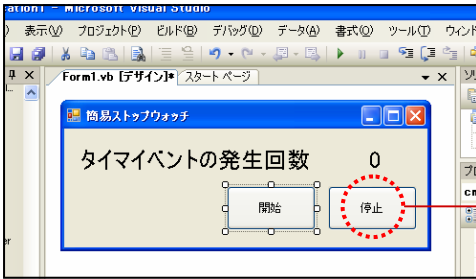
インテリセンス機能は、オブジェクト名のあとに『.(ピリオド)』を入力すると、その後に入力候補がリストから選べる機能です。今回は“.”の後にもEnableプロパティの『True/False』もリストから選べるようになっています。



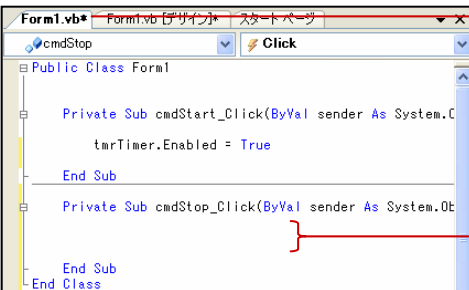
4-2-9.プログラム作成手順⑧ (タイマ停止処理の記述)

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。

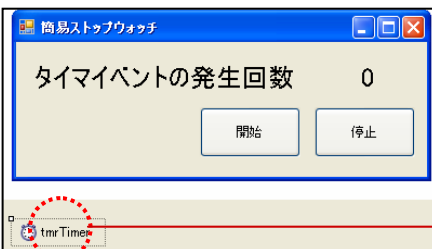
Private Sub cmdStop_Click () からEnd Subまでの間に、下記のコードを記述します。
『タイマコントロール (tmrTimer)』の『Enabledプロパティ』を『False (無効)』にするという意味です。

```
tmrTimer.Enabled = False
```

『タイマを停止します』

4-2-10.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)

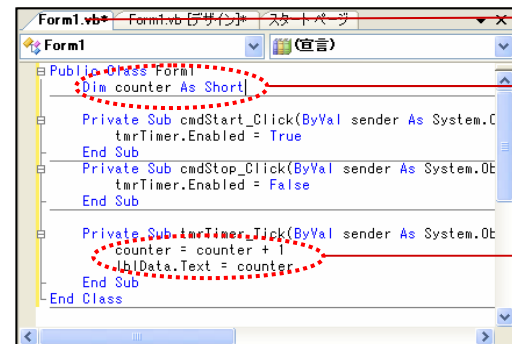
最後にタイマイベントが発生した際の処理を記述します。このタイマイベントは、タイマコントロールのEnabledプロパティが『True (有効)』の間、プロパティ『Interval』で設定した間隔毎に発生します。今回のプログラムでは、100msec (0.1秒) 毎に繰り返したい処理を記述します。タイマコントロール (tmrTimer) をダブルクリックして、コードウィンドウを開きます。



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_Tick () 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、Short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 Short

'カウントアップ用変数

最後にPrivate Sub tmrTimer_Tick () 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.Text = Counter

'変数に+1を行います
'変数を画面表示します

4-2-11. カウントアッププログラムリスト

Dim Counter As Short

'カウントアップ用変数

Private Sub cmdStart_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdStart.Click

tmrTimer.Enabled = True

'タイマを起動します

End Sub

Private Sub cmdStop_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdStop.Click

tmrTimer.Enabled = False

'タイマを停止します

End Sub

Private Sub tmrTimer_Tick (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tmrTimer.Tick

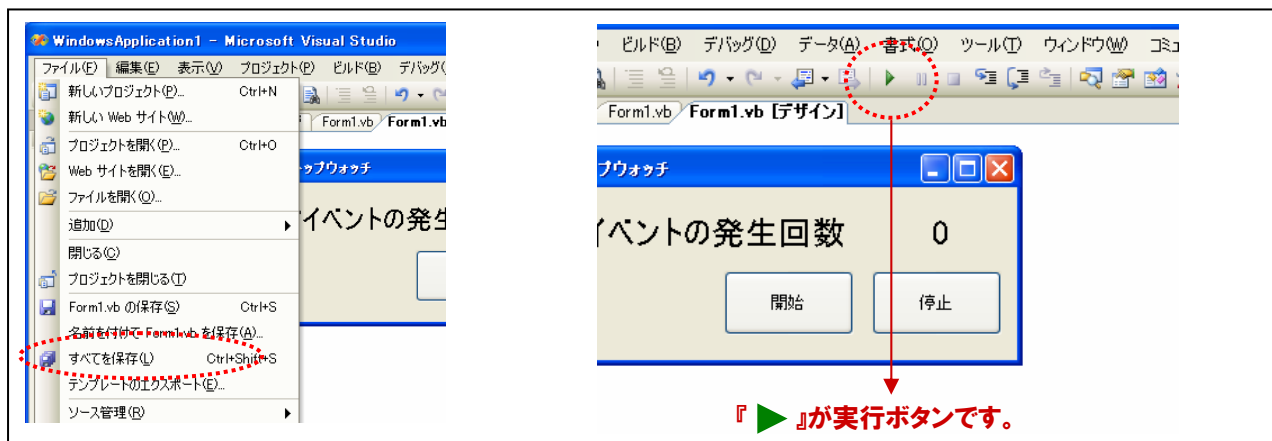
Counter = Counter + 1
lblData.Text = Counter

'変数に+1を行います
'変数を画面表示します

End Sub

4-2-12.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『すべてを保存』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。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

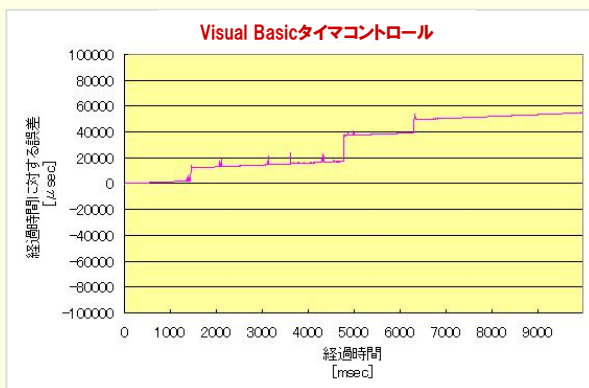
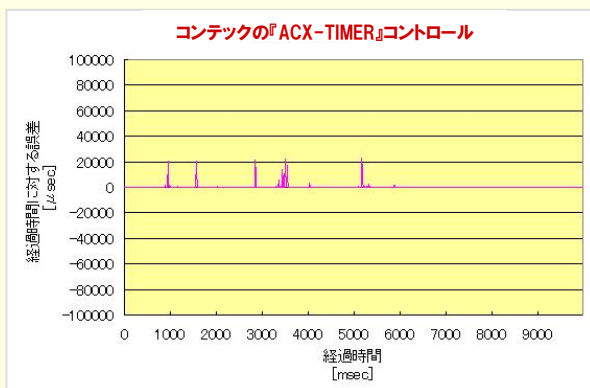
『Visual Basicのタイマコントロールと弊社提供のタイマコントロールとの違い』

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/>

4-3. タイマコントロールによる簡易連続アナログ入力プログラム

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、アナログ信号入力のプログラムを作成していきましょう。今回作成するプログラムは変換開始トリガおよび変換クロックは『ソフトウェア』です。

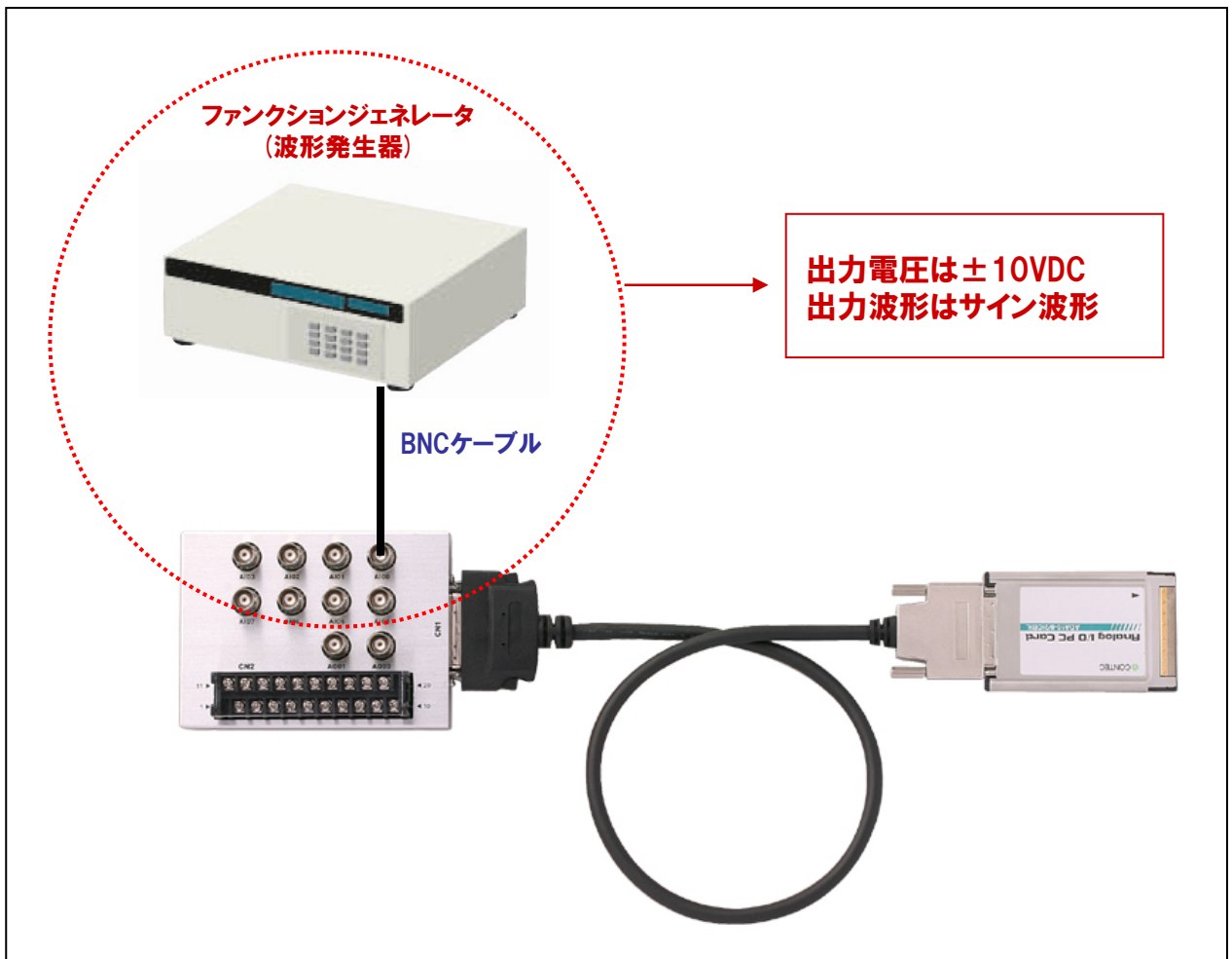
4-3-1. プログラム概要

タイマコントロール周期 (100msec) でアナログ入出力カード『ADA16-8/2 (CB) L』の『アナログ入力0ch』からデータを入力、画面に電圧値で表示します。

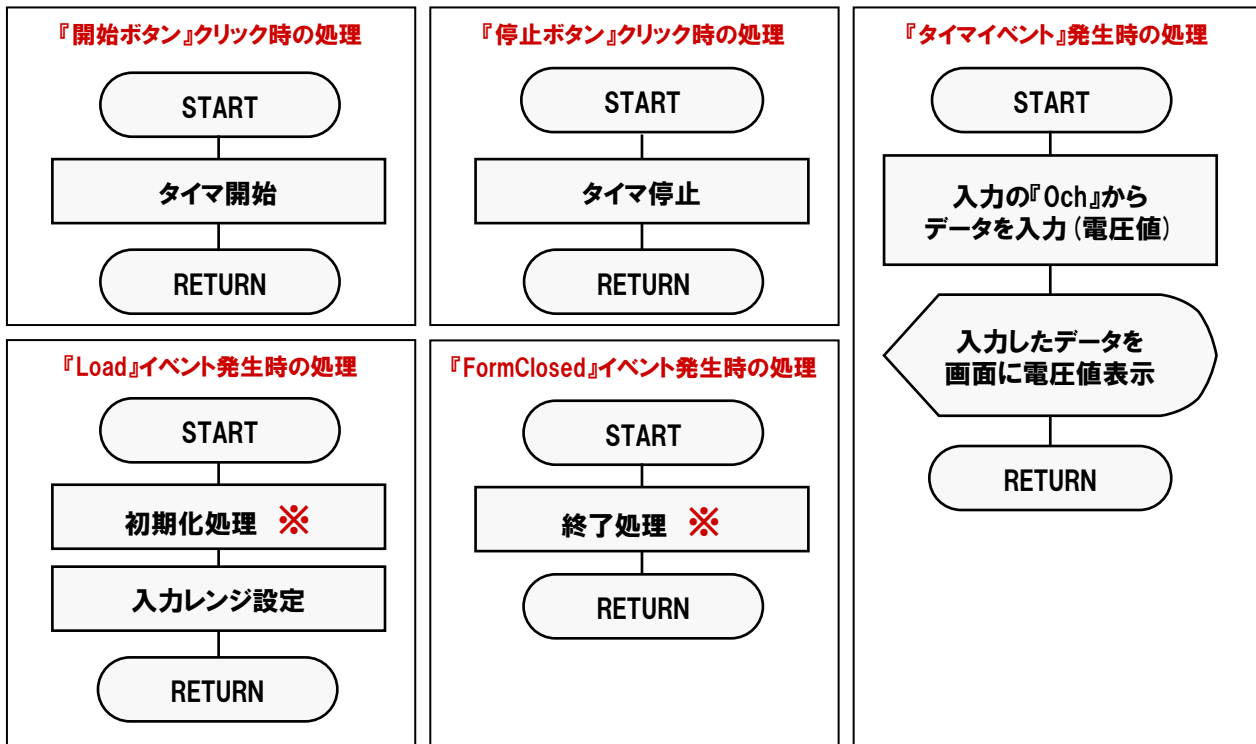


4-3-2. 実行環境

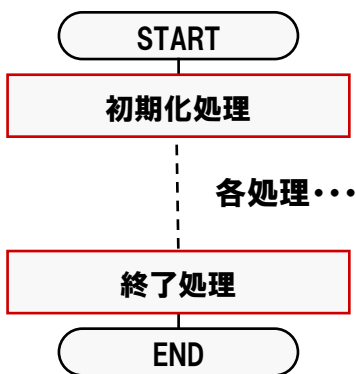
【3-6-3. 実習環境の構築】の環境を使用します。ファンクションジェネレータは、次の設定を事前に行っています。出力アナログ信号は±10VDC (VDC: 直流を示す)、出力波形はサイン波としています。



4-3-3.プログラム フローチャート



※: API-AIO (WDM) の処理体系は、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、その関数を実行します。



初期化処理で各種使用条件を設定します。この処理から、各関数が使用可能となります。

終了処理は、システム上変更した処理について、強制的に初期状態に戻します。

TOPICS

『API-AIO (WDM) について』

『API-AIO (WDM)』は、従来のアナログ入出力ドライバ『API-AIO (98/PC) **』と比較して、「より使いやすく便利に」、「より高性能に」を目指した、新アナログ入出力用ドライバです。ユーザーインターフェースの改善をはじめ、ハードウェアの機能をフルに活用することにより、アナログ入出力ボード/カードを使用したアプリケーション開発をより強力にサポートします。

- シンプルで使いやすい機能ごとに分類された関数を提供。
- 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。
- 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。

注1) 『API-AIO (WDM)』と従来ドライバ『API-AIO (98/PC) **』では互換性はありません。

注2) 使用するボード/カードによっては、『API-AIO (WDM)』のみのサポートとなります。

その他の違いは、API-AIO (WDM) のヘルプファイルを参照してください。ヘルプファイルには、これらの情報の他に 親切・丁寧な関数リファレンス、チュートリアルなどが収録されています。



『スタート』 - 『プログラム』 - 『CONTEC API-PAC (W32)』 - 『AIOWDM』 - 『API-AIO (WDM) HELP』

4-3-4.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)

[4-2-2.]~[4-2-7.]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 Windows Form titled '簡易アナログ入力' (Simple Analog Input) with the following controls and their properties:

- Form**
 - Text = 『簡易アナログ入力』
- Labelコントロール**
 - Name = 『IblData』
- 開始ボタン**
 - Text = 『開始』
 - Name = 『cmdStart』
- 停止ボタン**
 - Text = 『停止』
 - Name = 『cmdStop』
- タイマコントロール**
 - Name = 『tmrTimer』
 - Enabled = 『False』
 - Interval = 『100』
- Labelコントロール**
 - Name = 『IblTitle』
 - Text = 『入力電圧値 (V)』

4-3-5.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-AIO (WDM) 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。

- Visual Basicのメニュー『プロジェクト』 - 『既存項目の追加』を選択します。
- 標準設定でインストールした場合には下記の場合所に標準モジュールファイル (Caio.vb) 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) ¥AIOWDM¥Sample¥Inc¥Caio.vb

- ソリューション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。

The screenshots illustrate the following steps:

- Step 1:** The 'Project' menu is open, and 'Add Existing Item' is selected.
- Step 2:** The 'Add Existing Item' dialog is shown, with 'Caio.vb' selected in the file list.
- Step 3:** The 'Caio.vb' file is added to the project, and its code is visible in the Code window. The code includes declarations for API-AIO functions and timer events.

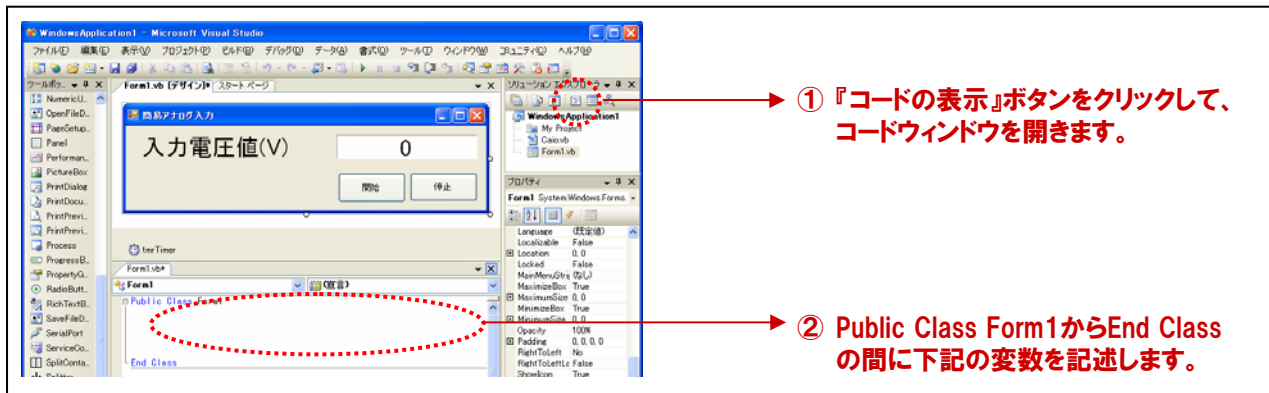
Caio.vbをダブルクリックで開くと、ドライバソフトウェアが提供している関数の情報(参照先)や、構造体の宣言、定数の宣言などがされているのが分かります。

4-3-6.プログラム作成手順③ (変数の追加)

簡易アナログ入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』 - 『プログラム』 - 『CONTEC API-PAC (W32)』 - 『AIOWDM』 - 『API-AIO (WDM) 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。

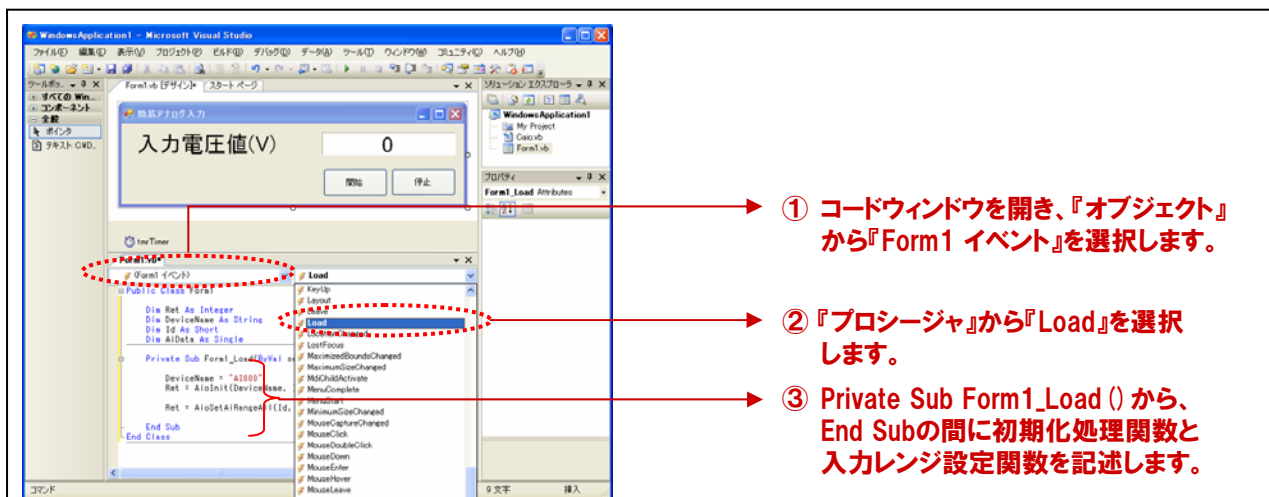
② Public Class Form1からEnd Classの間に下記の変数を記述します。

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

‘戻り値用変数
‘デバイス名設定用変数
‘デバイスID格納用変数
‘入力電圧値格納用変数

4-3-7.プログラム作成手順④ (初期化処理とアナログ入力初期設定の追加)

API-AIO (WDM) 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、変数を初期化の際に使われます。また、アナログ入力に関する初期設定 (本プログラムでは、入力レンジの設定のみ) を行います。



① コードウィンドウを開き、『オブジェクト』から『Form1 イベント』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 Form1_Load () から、End Subの間に初期化処理関数と入力レンジ設定関数を記述します。

下記のコードを『Private Sub Form1_Load ()』から『End Sub』の間に記述してください。

```
DeviceName = "AI0000"
Ret = AioInit (DeviceName, Id)
```

‘デバイス名を変数に格納
‘初期化処理 (デバイスハンドル取得)

```
Ret = AioSetAiRangeAll (Id, PM10)
```

‘入力レンジ設定 (入力レンジ±10V)

※:各関数で設定しているパラメータに関する詳細は、次項からの『関数リファレンス』で解説します。

初期化処理関数『AioInIt』リファレンス

■機能 デバイスファイルを作成し、以降デバイスを使用可能にします。デバイスにアクセスするには、まずこの関数を実行する必要があります。この関数がプロセスで初めて実行された時には、内部のパラメータがすべてデフォルト値に設定されます。デバイス内にレジスタを持つ場合はそのレジスタもデフォルト値に設定されます。AioExit関数を実行せずに続けてAioInItを実行しても、内部のパラメータはデフォルトに戻りません。内部パラメータをデフォルト値に戻すには、AioResetDeviceを使用します。

■書式 Visual Basic2005の場合

```
Dim Ret As Integer
Dim DeviceName AS String
Dim Id As Short
Ret = AioInIt ( DeviceName , Id )
```

■引数 DeviceName: デバイスマネージャのプロパティページで設定したデバイス名を指定します。

Id: ID (デバイスハンドル) を受け取る変数を指定します。以降の関数は、この変数に格納された値を用いてアクセスできます。

デバイス名に関して



3-5.手順 ④:ドライバソフトウェアの初期設定の項目で設定した、デバイス名を使用します。

本書では、デフォルト値『AI0000』を使用します。

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)。

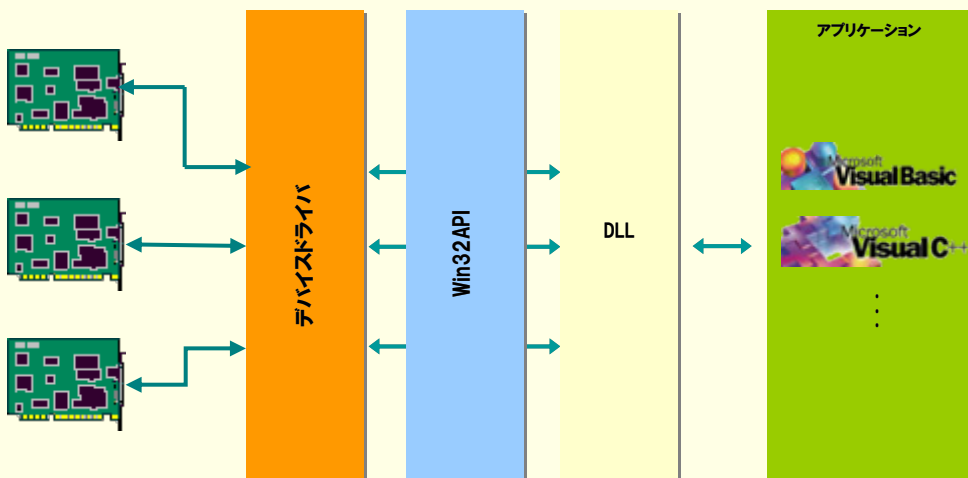
※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しています。実システムにおいては、関数を実行した後にエラー処理のコードを記述します。エラー処理の方法は、各サンプルプログラムを参照してください。

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がデバイスドライバ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。



全チャンネル入力レンジ設定関数『AioSetAiRangeAll』リファレンス

■機能 全チャンネルに対してアナログ入力レンジの設定を行います。

■書式 Visual Basic2005の場合

```
Dim Ret As Integer
Dim Id As Short
Dim AiRange As Short
Ret = AioSetAiRangeAll ( Id , AiRange )
```

■引数 Id : AiOnit 関数で取得したIDを指定します。

AiRange : アナログ入力レンジを以下の範囲からマクロ (標準モジュールファイル内で設定済みの定数) もしくは数値で指定します。設定できる値はデバイスにより異なります。

レンジ	マクロ	値
±10V	PM10	0
±5V	PM5	1
±2.5V	PM25	2
±1.25V	PM125	3
±1V	PM1	4
±0.625V	PM0625	5
±0.5V	PM05	6
±0.3125V	PM03125	7
±0.25V	PM025	8
±0.125V	PM0125	9
±0.1V	PM01	10
±0.05V	PM005	11
±0.025V	PM0025	12
±0.0125V	PM00125	13

レンジ	マクロ	値
0~10V	P10	50
0~5V	P5	51
0~4.095V	P4095	52
0~2.5V	P25	53
0~1.25V	P125	54
0~1V	P1	55
0~0.5V	P05	56
0~0.25V	P025	57
0~0.1V	P01	58
0~0.05V	P005	59
0~0.025V	P0025	60
0~0.0125V	P00125	61
0~20mA	P20MA	100
4~20mA	P4T020MA	101
1~5V	P1T05	150

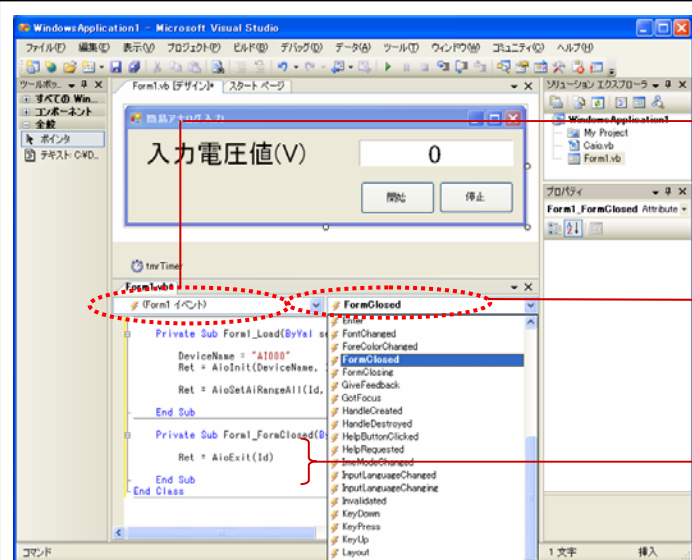
Ret: 終了情報 (戻り値) → 正常終了:0、エラー終了: 0以外 (詳細はヘルプの「戻り値一覧」参照)。

※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しています。
 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 本書にて使用している『ADA16-8/2 (CB) L』は、入力レンジは『±10V固定』です。

4-3-8.プログラム作成手順⑤ (終了処理の追加)

API-AIO (WDM) 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_FormClosedイベント』内で、終了処理関数を実行します。『Form_FormClosedイベント』は『Form』が画面から消去 (クローズド) 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『Form1 イベント』を選択します。

② 『プロシージャ』から『FormClosed』を選択します。

③ Private Sub Form1_FormClosed () 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 Form1_FormClosed ()』から『End Sub』の間に記述してください。

```
Ret = AioExit (Id)
```

‘終了処理 (デバイスハンドル開放)

終了処理関数 『AioExit』 リファレンス

■機能 ドライバの終了処理を行います。この関数は、アプリケーションの終了時に実行します。この関数を実行せずにアプリケーションを終了すると、以降デバイスにアクセスできなくなることがあります。

■書式 Visual Basic2005の場合

```
Dim Id As Short  
Dim Ret As Integer  
Ret = AioExit (Id)
```

■引数 Id: 終了するデバイスハンドルを指定します。AioInit関数で取得した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)。

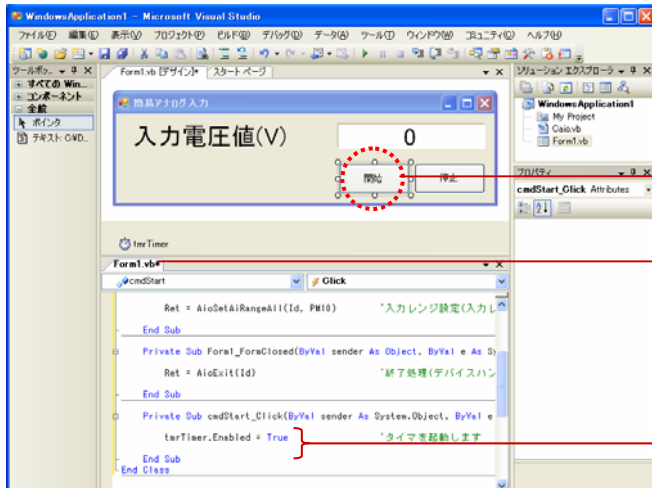
※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しています。実システムにおいては、関数を実行した後にエラー処理のコードを記述します。エラー処理の方法は、各サンプルプログラムを参照してください。

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

4-3-9.プログラム作成手順⑥ (タイマ開始処理の記述)

100msec (0.1秒) 毎にアナログ入力処理を行うために、【4-2.】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【4-2.】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。

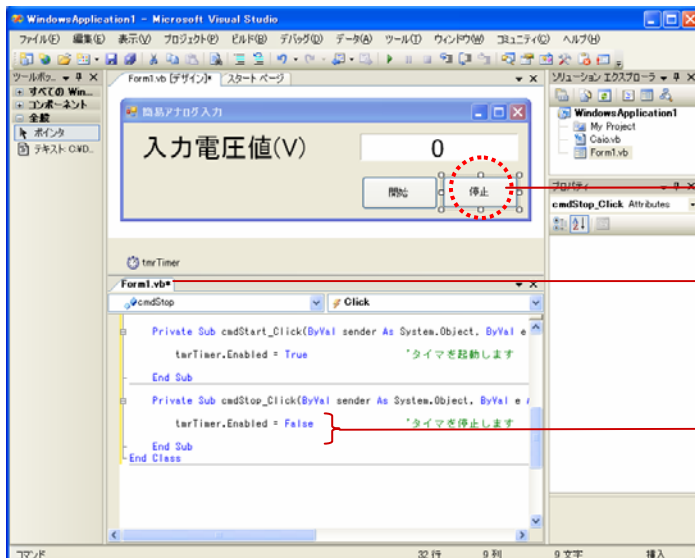
Private Sub cmdStart_Click () からEnd Subまでの間に、下記のコードを記述します。

```
tmrTimer.Enabled = True
```

‘タイマを起動します

4-3-10.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 () 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

4-3-11.プログラム作成手順⑧ (アナログ入力処理の追加)

アナログ入出力カードADA16-8/2 (CB) L (またはデモボード) の『入力0ch』から、アナログ (電圧) データの入力を行う処理を追加します。アナログ信号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2行を記述するだけで、100msec (0.1秒) 毎に指定したチャンネルからデータ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_Tick () から、End Subの間に記述します。
Labelコントロール (lblData) に入力0chのデータ『AiData』を表示していきます。データの後に"&" (文字列結合子) を使用して、文字列の"V"を付加しています。

Private Sub tmrTimer_Tick () から、End Subの間に下記2行を記述します。

```
Ret = AioSingleAiEx (Id, 0, AiData)      '入力0chのデータを変数AiDataに格納
lblData.Text = Format (AiData, "0.000000V") '変数AiDataを表示
```

アナログ入力 (電圧/電流値) 関数 『AioSingleAiEx』 リファレンス

- 機能 指定チャンネルを1回AD変換し、変換データを電圧値または電流値で返します。
- 書式 Visual Basic2005の場合

```
Dim Ret           As Integer
Dim AiData        As Single
Dim AiChannel     As Short
Dim Id            As Short

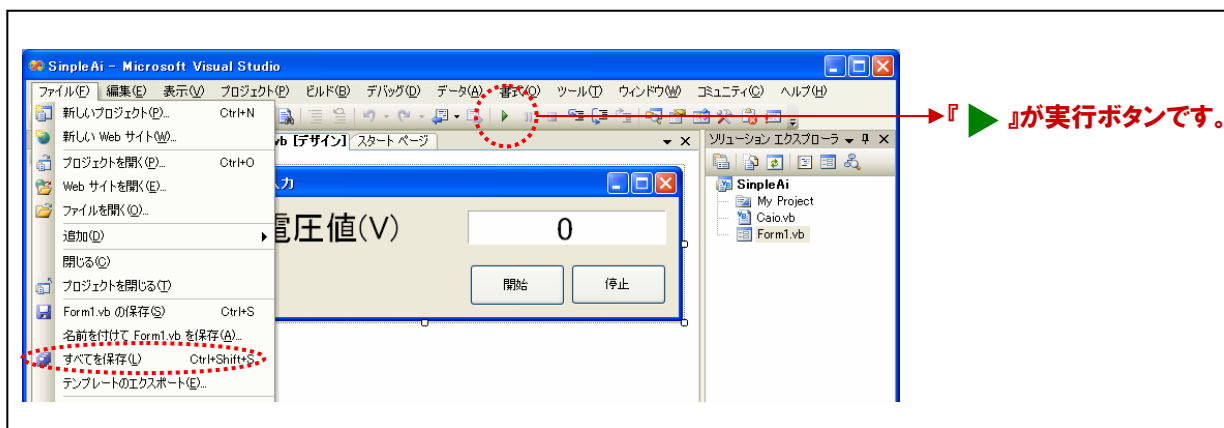
Ret = AioSingleAiEx ( Id , AiChannel , AiData )
```

- 引数 Id: AioInit関数で取得したIDを指定します。
- AiChannel: 変換するチャンネルを指定します。
- AiData: 変換データを格納する変数を指定します。変換データは電圧または電流値で格納されます。
- 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)。

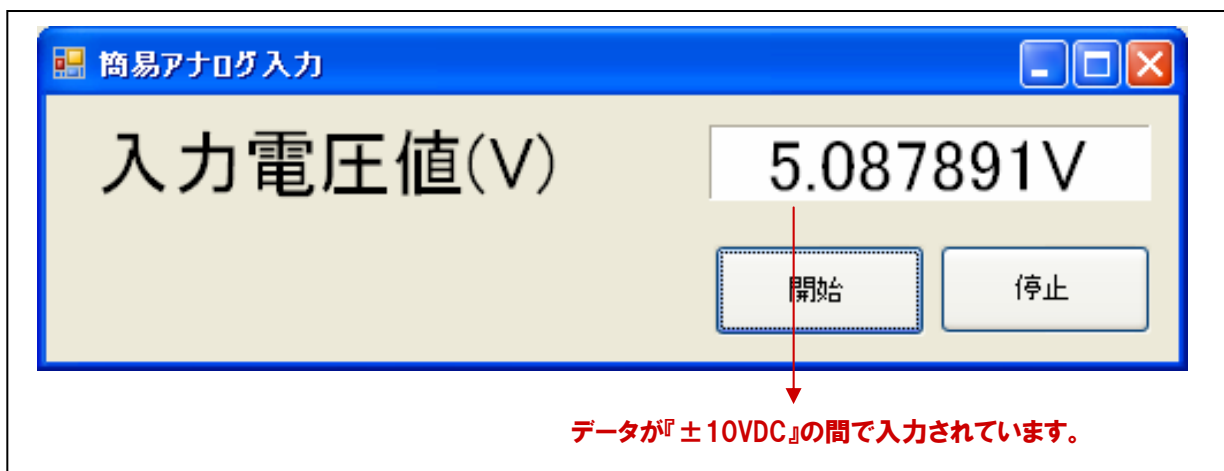
※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しています。実システムにおいては、関数を実行した後にエラー処理のコードを記述します。エラー処理の方法は、各サンプルプログラムを参照してください。

4-3-12.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。保存が終わりましたら、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 入力チャンネル0には、ファンクションジェネレータがBNCケーブルによって接続されています。ファンクションジェネレータの設定は、出力電圧=±10VDC、波形=サイン波です。画面上の『開始ボタン』をクリックして見ましょう。±10VDCの振幅で、データが増減して入力されていることがわかります。『停止ボタン』で入力が停止します。



入力されている電圧値をよく見ていると、“-10V”は正常に入力されているのに関わらず、“+”に関しては、約9.9997...Vまでしか入力されていません。その理由は【第2章 2-5-7.バイナリデータと電圧値の関係（分解能16ビットの場合）】で説明したとおりです。

アナログ入出力カード『ADA16-8/2 (CB) L』は、分解能16ビットのアナログ入出力が行えます。第2章で説明したとおり、アナログ入力ボード/カードは、アナログ信号をデジタル信号に変換しますが、アナログ信号をどの程度の細かさで表現できるかが分解能でした。『ADA16-8/2 (CB) L』は、あるアナログ信号を16ビット、すなわち65536 (2^{16}) の細かさで分解し、その結果を入力してくれます。そして、本来アナログボード/カードから入力される値は、その分解された『バイナリ(2進数)』のデータなのです。【4-3-11】で使用したアナログ入力関数『AioSingleAiEx』は、本来入力されている16ビットのバイナリデータ『0000~FFFF (16進数)』をボード/カードの分解能と入力レンジを元に適切な電圧値(電流値)に変換して、入力してくれていたのです。

API-AIO (WDM) には、『AioSingleAi』という関数も提供されています。指定チャンネルを1回A/D変換し、変換データをバイナリ値で返す機能を持った関数です。

アナログ入力関数『AioSingleAi』リファレンス

- 機能 指定チャンネルを1回AD変換し、変換データをバイナリ値で返します。
- 書式 Visual Basic2005の場合

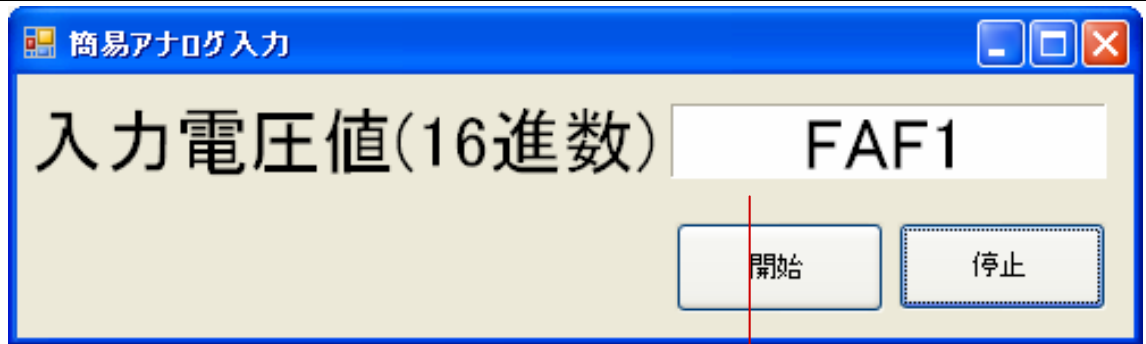
```
Dim Ret As Integer
Dim AiData As Integer
Dim AiChannel As Short
Dim Id As Short

Ret = AioSingleAi ( Id , AiChannel , AiData )
```

- 引数 Id: AiInIt関数で取得したIDを指定します。
- AiChannel: 変換するチャンネルを指定します。
- AiData: 変換データを格納する変数を指定します。変換データはバイナリデータです。
- 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、割愛しています。
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。

- ③ 実際に関数を入れ替えて実行、データを16進数で表示させると、データは『0000~FFFF』になります。



データが『0000~FFFF』の間で入力されています。

API-AIO (WDM) が提供している『AioSingleAiEx』のような高機能関数が提供されていない場合には、入力されているバイナリ(2進数)データから、公式を使用して適切な電圧/電流値に変換します。

バイポーラ(双極)レンジの場合

$$\text{電圧 (V)} = \frac{\text{デジタル値}}{\text{分解能}} \times \text{フルスケールレンジ} - \frac{\text{フルスケールレンジ}}{2}$$

- ※ フルスケールレンジ: ±10Vレンジの場合は『20』。
- ※ 『デジタル値』は、入力ポートから読み込んだバイナリデータ。
- ※ 分解能は、12ビットの場合は『4096』、16ビットの場合は『65536』。

ユニポーラ(単極)レンジの場合

$$\text{電圧 (V)} = \frac{\text{デジタル値}}{\text{分解能}} \times \text{フルスケールレンジ}$$

- ※ フルスケールレンジ: 0~10Vレンジの場合は『10』。
- ※ 『デジタル値』は、入力ポートから読み込んだバイナリデータ。
- ※ 分解能は、12ビットの場合は『4096』、16ビットの場合は『65536』。

4-3-13. タイマコントロールによる簡易連続アナログ入力プログラムリスト

Dim Ret	As Integer	‘戻り値用変数
Dim DeviceName	As String	‘デバイス名設定用変数
Dim Id	As Short	‘デバイスID格納用変数
Dim AiData	As Single	‘入力電圧値格納用変数

Private Sub Form1_Load (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

DeviceName = "AI0000"	‘デバイス名を変数に格納
Ret = AiInIt (DeviceName, Id)	‘初期化処理 (デバイスハンドル取得)
Ret = AioSetAiRangeAll (Id, PM10)	‘入力レンジ設定 (入力レンジ±10V)

End Sub

Private Sub Form1_FormClosed (ByVal sender As Object, ByVal e As System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed

Ret = AioExit (Id)	‘終了処理 (デバイスハンドル開放)
--------------------	--------------------

End Sub

Private Sub cmdStart_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdStart.Click

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

End Sub

Private Sub cmdStop_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdStop.Click

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

End Sub

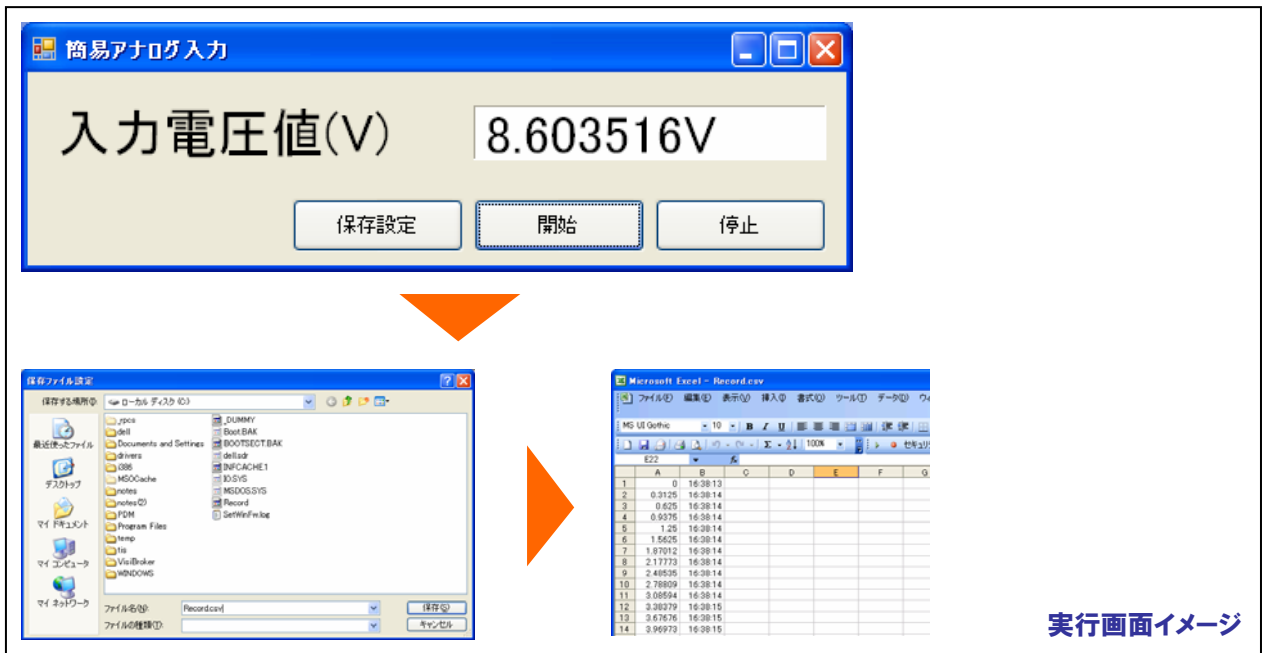
Private Sub tmrTimer_Tick (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tmrTimer.Tick

Ret = AioSingleAiEx (Id, 0, AiData)	‘入力0chのデータを変数AiDataに格納
lblData.Text = Format (AiData, "0.000000V")	‘変数AiDataを表示

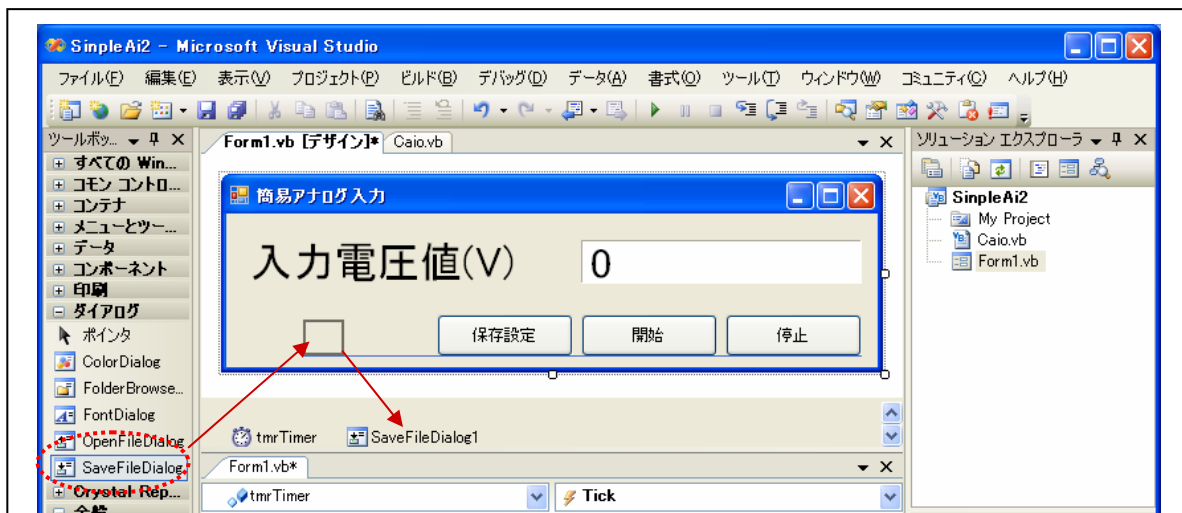
End Sub

4-3-14.プログラムの改造 (データロギング (ファイル保存) 機能の付加)

作成した簡易アナログ入力プログラムでは、入力されているデータは画面表示のみでした。このプログラムを Visual Basic2005の機能を使用して、データのファイル保存ができるように改造してみましょう。



- ① Visual Basic2005には、各種ダイアログボックスを簡単に表示させるためのコントロールが提供されています。コントロールツールボックスから『SaveFileDialog』を選択（クリック）して、フォームに貼り付けます。なお、本コントロールは、プログラム実行時には不可視となるため、適当な場所に貼り付けると自動で枠外に設置されます。



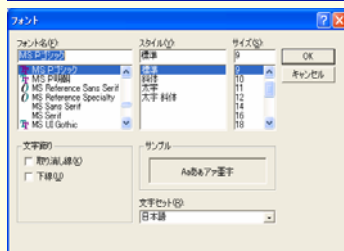
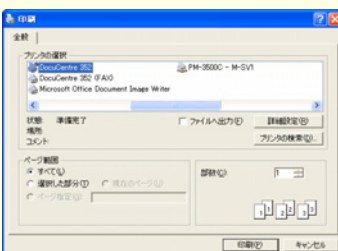
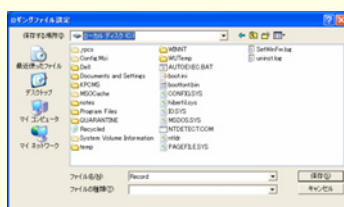
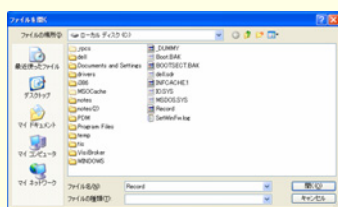
適当な場所に貼り付けると、自動で枠外に設置されます。

TOPICS:『CommonDialog Controlについて』

CommonDialog Controlは、ファイルのオープン、保存、印刷、色やフォントの選択など、Windowsアプリケーションで頻繁に使用するダイアログボックスを簡単に表示できる機能を持ったコントロール（ソフトウェア部品）です。Visual Basic2005では以下の8種類のダイアログコントロールが標準で使用できます。

■コモンダイアログの種類

- | | |
|--------------|-----------------------------|
| ①ファイルを開く | → OpenFileDialogを使用します。 |
| ②ファイル名を付けて保存 | → SaveFileDialogを使用します。 |
| ③色の指定 | → ColorDialogを使用します。 |
| ④フォントの指定 | → FontDialogを使用します。 |
| ⑤フォルダの指定 | → FolderBrowseDialogを使用します。 |
| ⑥印刷設定 | → PrintDialogを使用します。 |
| ⑦印刷プレビュー | → PrintPreviewDialogを使用します。 |
| ⑧ページ設定 | → PageSetupDialogを使用します。 |



② 次に、ファイル保存ダイアログボックスを表示させるための、コマンドボタンをフォームに追加し、以下のとおりにプロパティ設定をおこないます。

The screenshot shows the Visual Studio IDE with a form titled 'Form1.vb' and a 'SaveFileDialog1' control. A red dashed circle highlights the '保存設定' button on the form. Another red dashed circle highlights the 'SaveFileDialog1' control in the Properties window. Red arrows point from these circles to the corresponding text boxes on the right.

■ 保存ファイル設定ボタン

- Text = 『保存設定』
- Name = 『cmdFile』

■ セーブファイルダイアログコントロール

- Name = 『SaveFileDialog1』
- DefaultExt = 『csv』
(DefaultExt: 既定の拡張子)
- FileName = 『Record』
(FileName: 既定のファイル名)
- Title = 『保存ファイル設定』
(Title: ファイル保存ダイアログのタイトル)

プロパティ

SaveFileDialog1 System.Windows.Fc

ApplicationSettings	
(Name)	SaveFileDialog1
AddExtension	True
CheckFileExists	False
CheckPathExists	True
CreatePrompt	False
DefaultExt	csv
DereferenceLink	True
FileName	Record
Filter	
FilterIndex	1
GenerateMember	True
InitialDirectory	
Modifiers	Friend
OverwritePrompt	True
RestoreDirectory	False
ShowHelp	False
SupportMultiDott	False
Tag	
Title	保存ファイル設定
ValidateNames	True

FileName
ダイアログ ボックスで初めて表示されるファイル、またはユーザーにより選択された最後のファイルです。

- ③ 保存設定ボタンをクリックした時の処理を記述します。行う処理は、ファイル保存ダイアログボックスの表示とファイルを作成する場所の指定です。

① 『保存設定』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 cmdFile_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 cmdFile_Click () からEnd Subまでの間に、下記のコードを記述します。

Dim SaveFileName As String	‘ファイルパス情報格納用変数
SaveFileDialog1.ShowDialog	‘ファイルセーブダイアログ表示
SaveFileName = SaveFileDialog1.FileName	‘選択したファイルを変数に格納
FileOpen (1,SaveFileName,OpenMode.Output,OpenAccess.Write)	‘ファイルオープン

- 解説:**
- ① ファイルパス情報を保存するための変数 (文字型) を宣言します。
 - ② セーブファイルダイアログボックスの『ShowDialog』メソッドを使用して、ファイルの保存ダイアログを表示させます。
 - ③ セーブファイルダイアログボックスのFileNameプロパティには、ダイアログで指定した場所のファイルの絶対パス情報が入力されているので、その情報を変数SaveFileNameに保存します。
 - ④ 指定した場所へ書き込み専用にてファイルをオープン (作成) します。

TOPICS:『Visual Basic2005でのファイルのオープン/書き込み/クローズの方法』

Visual Basic2005でのファイル書き込み操作には、以下の3つの手順が必要となります。

■ ファイルのオープン:[FileOpenステートメント]

例:C:¥ContecにTest.csvという書き込み専用のファイルを“1”という番号で開く場合。

FileOpen (1,"C:¥Contec¥Test.csv"OpenMode.Output,OpenAccess.Write)

■ ファイルへの書き込み:[Printステートメント]

例:"1"という番号のファイルに現在時刻を書き込む場合。

Print (1,Now)

■ ファイルのクローズ:[FileCloseステートメント]

例:"1"という番号のファイルを閉じる。

FileClose (1)

- ④ 最後に、データのファイルへの書き込みと、ファイルのクローズ処理の記述を行います。ファイルの書き込みは、タイマイベント処理部、ファイルのクローズは停止ボタンクリック時処理部にて行います。
下記のとおり、コードを追加してください。

Dim Ret	As Integer	‘戻り値用変数
Dim DeviceName	As String	‘デバイス名設定用変数
Dim Id	As Short	‘デバイスID格納用変数
Dim AiData	As Single	‘入力電圧値格納用変数

```
Private Sub Form1_Load (ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
```

DeviceName = "AIO000"	‘デバイス名を変数に格納
Ret = AioInit (DeviceName, Id)	‘初期化処理 (デバイスハンドル取得)
Ret = AioSetAiRangeAll (Id, PM10)	‘入力レンジ設定 (入力レンジ±10V)

```
End Sub
```

```
Private Sub Form1_FormClosed (ByVal sender As Object, ByVal e As
System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
```

Ret = AioExit (Id)	‘終了処理 (デバイスハンドル開放)
--------------------	--------------------

```
End Sub
```

```
Private Sub cmdStart_Click (ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles cmdStart.Click
```

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

```
End Sub
```

```
Private Sub cmdStop_Click (ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles cmdStop.Click
```

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

FileClose (1)	‘ファイルを閉じる
---------------	-----------



```
End Sub
```

```
Private Sub tmrTimer_Tick (ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles tmrTimer.Tick
```

Ret = AioSingleAiEx (Id, 0, AiData)	‘入力0chのデータを変数AiDataに格納
IblData.Text = Format (AiData, "0.000000V")	‘変数AiDataを表示

Print (1,Format (AiData,"0.00000"), ",," , Now,vbCrLf)	‘入力電圧をファイルに書き込む
--	-----------------



```
End Sub
```

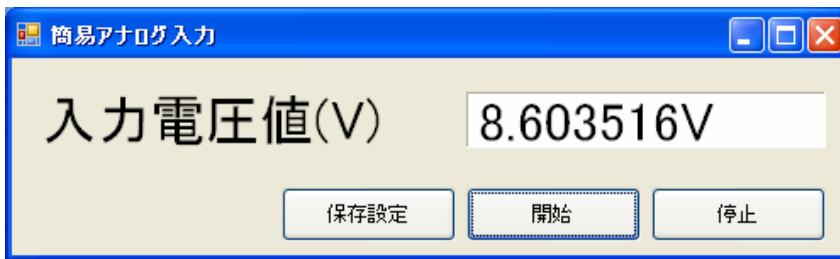
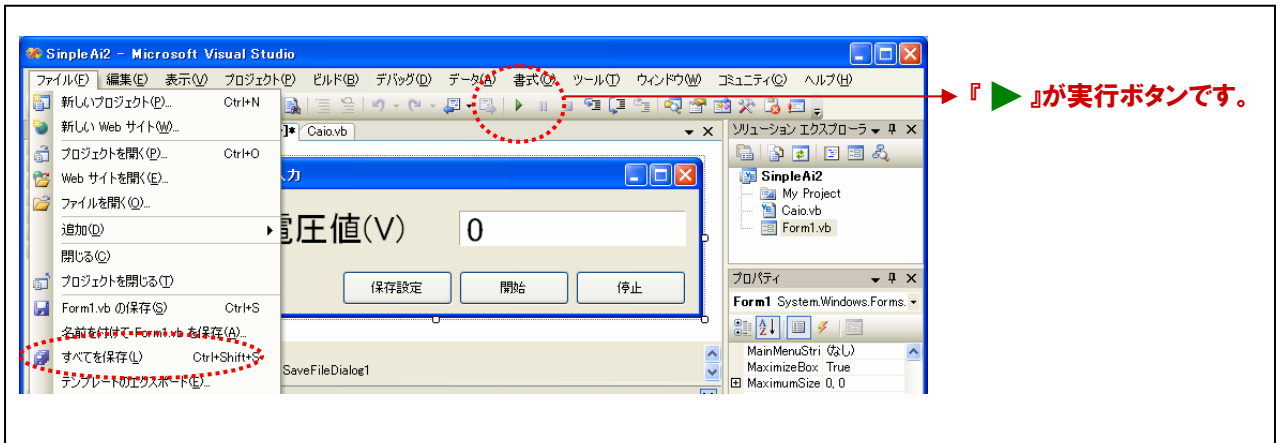
```
Private Sub cmdFile_Click (ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles cmdFile.Click
```

Dim SaveFileName As String	‘ファイルパス情報格納用変数
SaveFileDialog1.ShowDialog	‘ファイルセーブダイアログ表示
SaveFileName = SaveFileDialog1.FileName	‘選択したファイルを変数に格納
FileOpen (1,SaveFileName,OpenMode.Output,OpenAccess.Write)	‘ファイルオープン



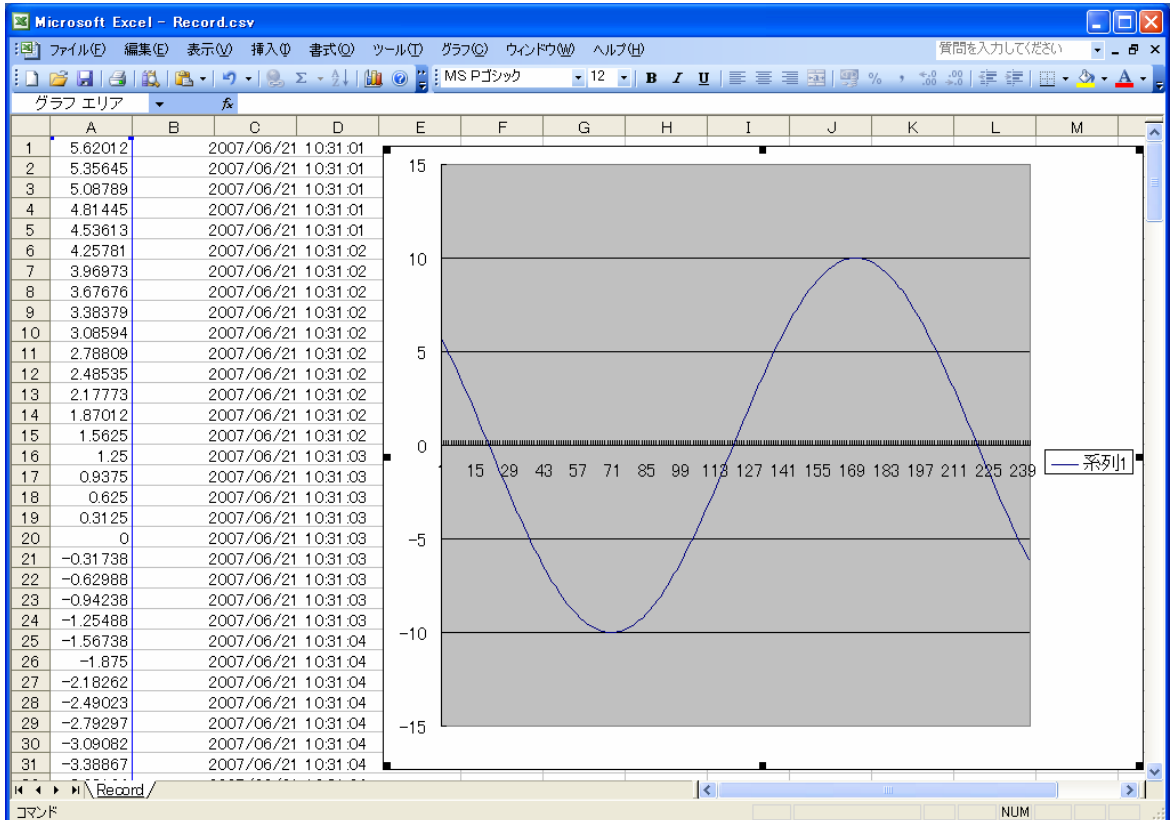
```
End Sub
```


- ⑤ プロジェクトを保存した後、プログラムを実行してみましょ。この際、必ず『保存設定』ボタンを最初にクリックして、データを書き込むファイルを任意の場所に指定してから『開始』ボタンで計測開始してください。



データの表示と同時に『ファイルの保存ダイアログボックス』で指定した場所に作られた”Record.csv”というファイルにデータが書き込まれていきます。停止ボタンをクリックして、プログラムを停止します。

“csv”はカンマ区切り形式のテキストファイルなので、Microsoft Excelでも開くことができます。データの確認だけでなく、Microsoft Excelのグラフ機能を使用すればデータのグラフ化も容易に実現可能です。



4-4.FIFOメモリを使用した高速サンプリング (簡易オシロスコープ：データ表示)

【4-3.】では、Visual Basic2005のタイマコントロール使用し、リアルタイムにデータ入力するプログラムを作成しましたが【4-2.】のタイマプログラムで説明したとおり、高速なサンプリングはできません。データを高速 (μsec 単位) でサンプリングするためには、デバイスに搭載されているバッファメモリにデータを保存し、サンプリング終了後にパソコンに取り込む方法があります。その際のクロックとして内部 (デバイスに搭載されているタイマカウンタ) クロックを使用すれば、高速・高精度なサンプリングが可能になります。ここでは、内部クロックとデバイスに搭載されているバッファメモリを使用した高速サンプリングプログラム (簡易オシロスコープ) を作成していきます。

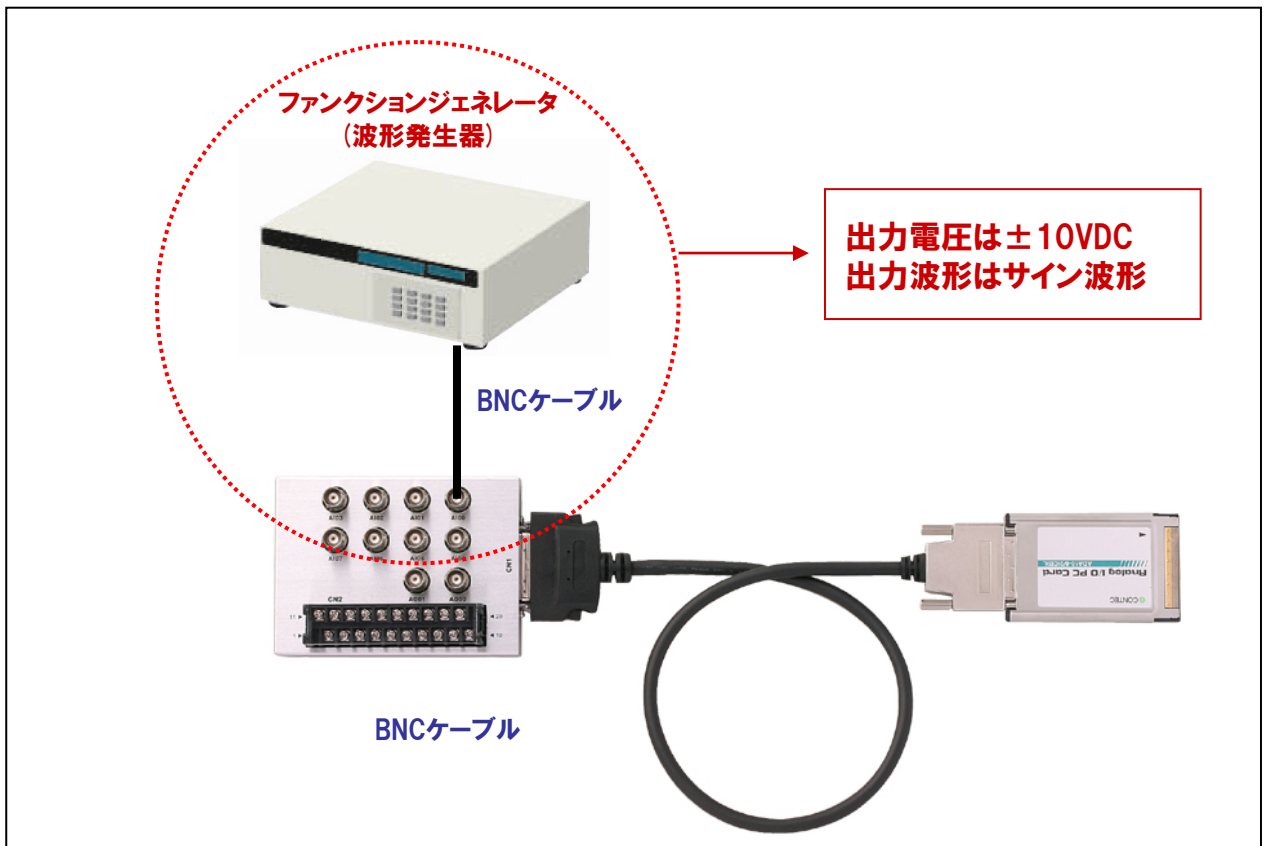
4-4-1.プログラム概要

バッファメモリ (FIFO形式) を使用して、 $1000\ \mu\text{sec}$ 周期のデータを1000回サンプリングし、テキストボックスに表示するプログラムです。アナログ入力カード『ADA16-8/2 (CB) L』の『アナログ入力0ch』のみ使用します。

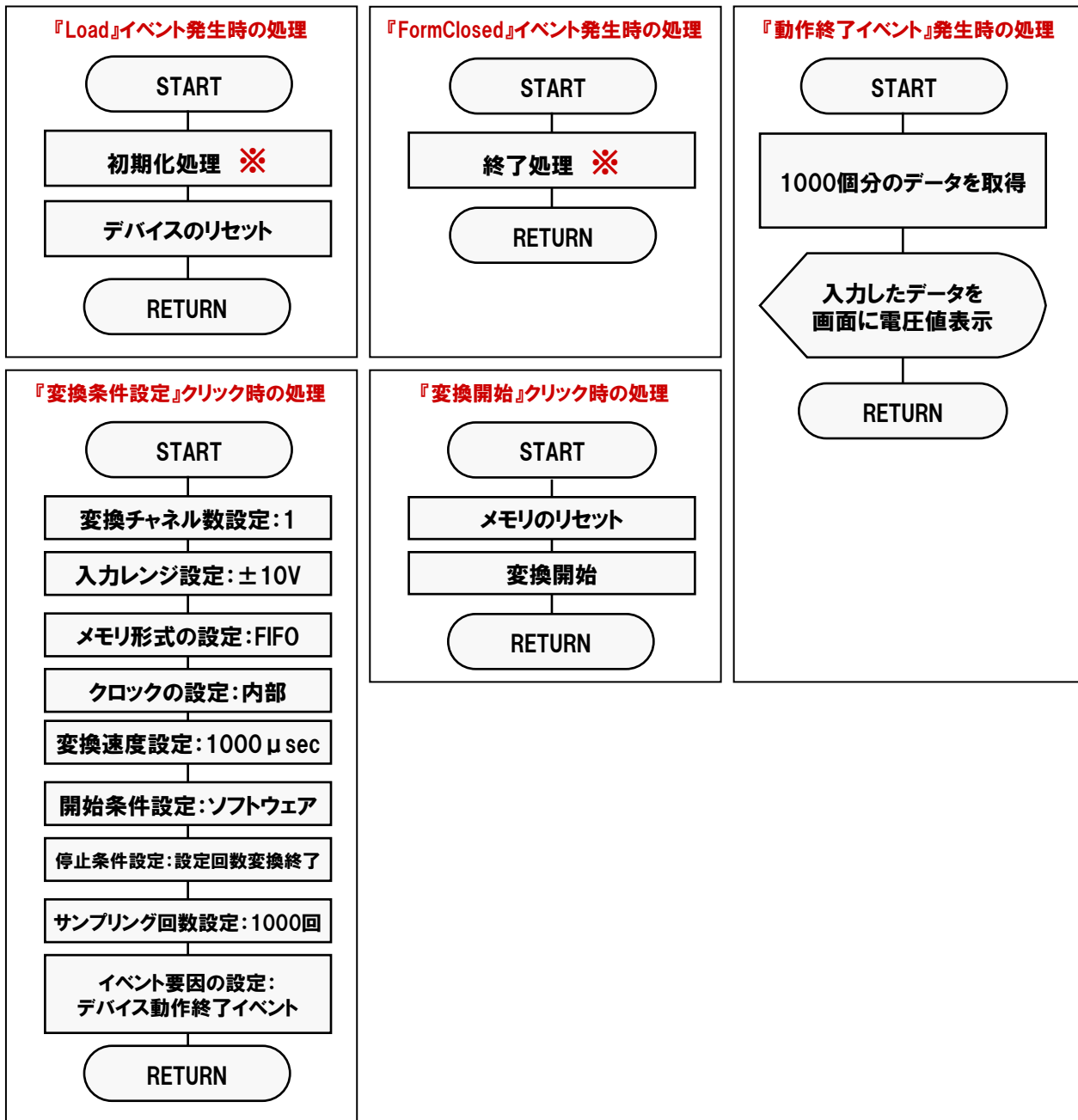


4-4-2.実行環境

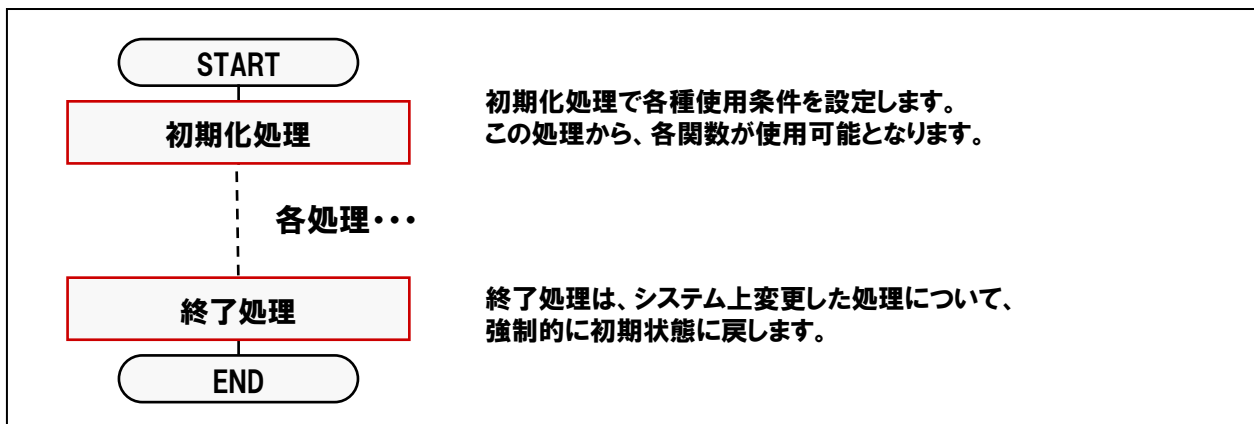
【3-6-3.実習環境の構築】の環境を使用します。ファンクションジェネレータは、次の設定を事前に行っています。出力アナログ信号は $\pm 10\text{VDC}$ (VDC:直流を示す)、出力波形はサイン波としています。



4-4-3.プログラム フローチャート



※: API-AIO (WDM) の処理体系は、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、その関数を実行します。



4-4-4.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)

[4-2-2.]~[4-2-7.]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**
○Text = 『FIFOを使用した簡易オシロスコープ』

■ **変換条件設定ボタン**
○Text = 『変換条件設定』
○Name = 『cmdSet』

■ **テキストボックスコントロール**
○Name = 『txtData』
○MultiLine = 『True』
○ScrollBars = 『Vertical』

■ **変換開始ボタン**
○Text = 『変換開始』
○Name = 『cmdStart』

4-4-5.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-AIO (WDM) 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』 - 『既存項目の追加』を選択します。
- ② 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 (Caio.vb) 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) ¥AIOWDM¥Sample¥Inc¥Caio.vb

- ③ ソリューション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

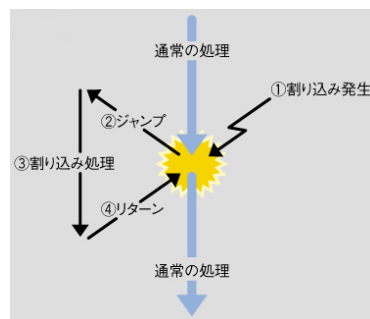
Caio.vbをダブルクリックで開くと、ドライバソフトウェアが提供している関数の情報(参照先)や、構造体の宣言、定数の宣言などがされているのが分かります。

4-4-6.プログラム作成手順③ (割り込み処理の実現)

【2-5-13.】にて説明した『割り込み処理』に関して、ソフトウェアに重点をおいた説明をしておきます。

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。



② Windowsにおける割り込み処理

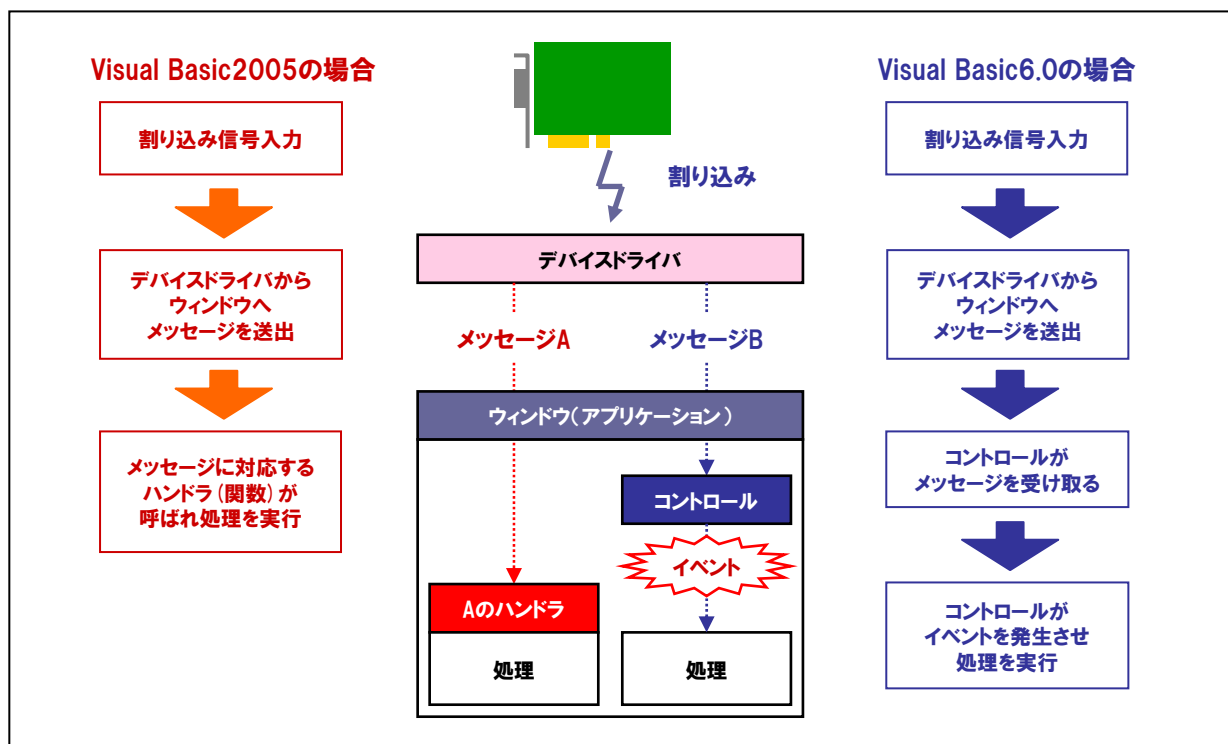
Windowsでは、外部機器から割り込み信号が入力されると、『デバイスドライバ』にその旨が通知されます。通知を受けた『デバイスドライバ』は『メッセージ』と呼ばれる32ビットの数値をアプリケーションに発信します。アプリケーションはそれらの送られてきた『メッセージ』に対応した各処理を行うことによって動作しています。

(a) 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められています (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。

(b) Visual Basic6.0以前とVisual Basic2005における割り込み入力処理

Visual Basic6.0以前は、Visual Basic2005と異なり、この『メッセージ』を直接受け取ることができませんでした。そのため、API-AIO (WDM) 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、『CONTEC Message Control: Cmessageコントロール』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『Cmessageコントロール』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。Visual Basic2005では、メッセージに対応した関数であるWndProcが呼ばれ処理を実行します。



③ API-AIO (WDM) におけるWndProcの役割

WndProcは、Windowsメッセージと内容を取得する関数です。Visual Basic2005、Visual C#で使します。API-AIO (WDM) では、以下のメッセージを扱います。

アナログ入力メッセージ要因	マクロ	値
AD変換開始条件成立イベント	AIOM_AIE_START	1000H
リピート終了イベント	AIOM_AIE_RPTEND	1001H
デバイス動作終了イベント	AIOM_AIE_END	1002H
指定サンプリング回数格納イベント	AIOM_AIE_DATA_NUM	1003H
指定転送数毎イベント	AIOM_AIE_DATA_TSF	1007H
オーバーフローイベント	AIOM_AIE_OFERR	1004H
サンプリングクロックエラーイベント	AIOM_AIE_SCERR	1005H
A/D変換エラーイベント	AIOM_AIE_ADERR	1006H
アナログ出力メッセージ要因	マクロ	値
DA変換開始条件成立イベント	AIOM_AOE_START	1020H
リピート終了イベント	AIOM_AOE_RPTEND	1021H
デバイス動作終了イベント	AIOM_AOE_END	1022H
指定転送数毎イベント	AIOM_AOE_DATA_TSF	1027H
指定サンプリング回数出力イベント	AIOM_AOE_DATA_NUM	1023H
サンプリングクロックエラーイベント	AIOM_AOE_SCERR	1025H
D/A変換エラーイベント	AIOM_AOE_ADERR	1026H
カウンタメッセージ要因	マクロ	値
動作終了イベント	AIOM_CNTE_END	1040H
動作開始条件成立イベント	AIOM_CNTE_START	1041H
比較カウント一致イベント	AIOM_CNTE_DATA_NUM	1042H
カウントオーバーランイベント	AIOM_CNTE_ORERR	1043H
タイマメッセージ要因	マクロ	値
インターバル成立イベント	AIOM_TME_INT	1060H

Visual Basic2005でのイベントメッセージルーチンは次の書式になります。

`Protected Overrides Sub WndProc (ByRef m As System.Windows.Forms.Message)`

- ◎m.Msg : メッセージ番号が渡されます。
- ◎m.WParam : 下位2バイトにIDが渡されます。上位2バイトは現在使用しません。
- ◎m.LParam : イベントごとに固有のパラメータが渡されます。本書では『デバイス動作終了イベント』を使用しますので、下表のとおり『m.LParam』引数には『現在のサンプリング回数』が渡されてきます。

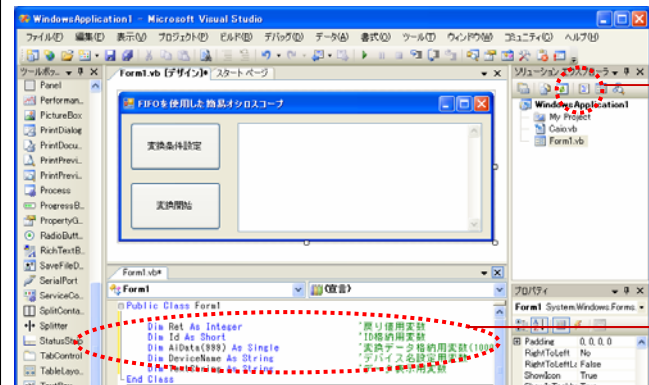
イベント要因	パラメータ
AD変換開始条件成立イベント	なし
リピート終了イベント	現在のリピート回数
デバイス動作終了イベント	現在のサンプリング回数
指定サンプリング回数出力イベント	現在のサンプリング回数
指定転送数毎イベント	現在の転送回数
オーバーフローエラーイベント	現在のサンプリング回数
サンプリングクロックエラーイベント	現在のサンプリング回数
AD変換エラーイベント	現在のサンプリング回数

4-4-7.プログラム作成手順④ (変数の追加)

簡易オシロスコープ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』 - 『プログラム』 - 『CONTEC API-PAC (W32)』 - 『AIOWDM』 - 『API-AIO (WDM) 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。

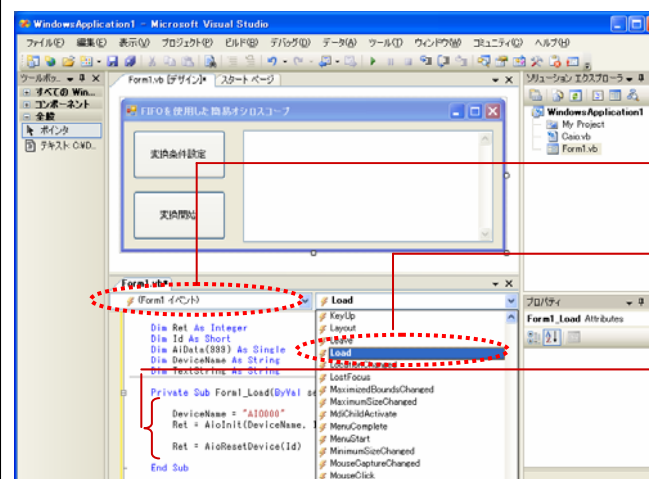
② Public Class Form1からEnd Classの間に下記の変数を記述します。

Dim Ret	As Integer	‘戻り値用変数
Dim Id	As Short	‘ID格納用変数
Dim AiData (999)	As Single	‘変換データ格納用変数 (1000個分の配列変数)
Dim DeviceName	As String	‘デバイス名設定用変数
Dim TextString	As String	‘データ表示用変数

4-4-8.プログラム作成手順⑤ (初期化処理とアナログ入力デバイスリセットの追加)

API-AIO (WDM) 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。

また、アナログ入力デバイスのリセットを行います。



① コードウィンドウを開き、『オブジェクト』から『Form1 イベント』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 Form1_Load () から、End Subの間に初期化処理関数とデバイスリセット関数を記述します。

下記のコードを『Private Sub Form1_Load ()』から『End Sub』の間に記述してください。

DeviceName = "AIO000"	‘デバイス名を変数に格納
Ret = AioInit (DeviceName, Id)	‘初期化処理 (デバイスハンドル取得)
Ret = AioResetDevice (Id)	‘指定デバイスのリセット

※:各関数で設定しているパラメータに関する詳細は、次項からの『関数リファレンス』で解説します。

初期化処理関数『AioInIt』リファレンス

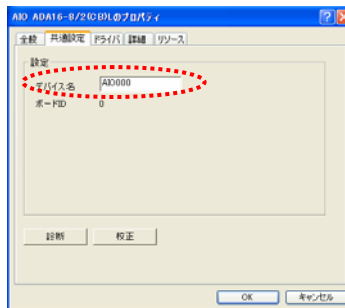
- 機能 デバイスファイルを作成し、以降デバイスを使用可能にします。デバイスにアクセスするには、まずこの関数を実行する必要があります。この関数がプロセスで初めて実行された時には、内部のパラメータがすべてデフォルト値に設定されます。デバイス内にレジスタを持つ場合はそのレジスタもデフォルト値に設定されます。AioExit関数を実行せずに続けてAioInItを実行しても、内部のパラメータはデフォルトに戻りません。内部パラメータをデフォルト値に戻すには、AioResetDeviceを使用します。

- 書式 Visual Basic2005の場合

```
Dim Ret As Integer
Dim DeviceName As String
Dim Id As Short
Ret = AioInIt ( DeviceName , Id )
```

- 引数 DeviceName: デバイスマネージャのプロパティページ、または設定ツールで設定したデバイス名を指定します。
Id: ID (デバイスハンドル)を受け取る変数を指定します。以降の関数は、この変数に格納された値を用いてアクセスできます。

デバイス名に関して



3-5.手順④:ドライバソフトウェアの初期設定の項目で設定した、デバイス名を使用します。

本書では、デフォルト値『AI0000』を使用します。

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、割愛しています。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。

デバイスリセット処理関数『AioResetDevice』リファレンス

- 機能 デバイスのリセット、ドライバの初期化を行います。

- 書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Ret = AioResetDevice ( Id )
```

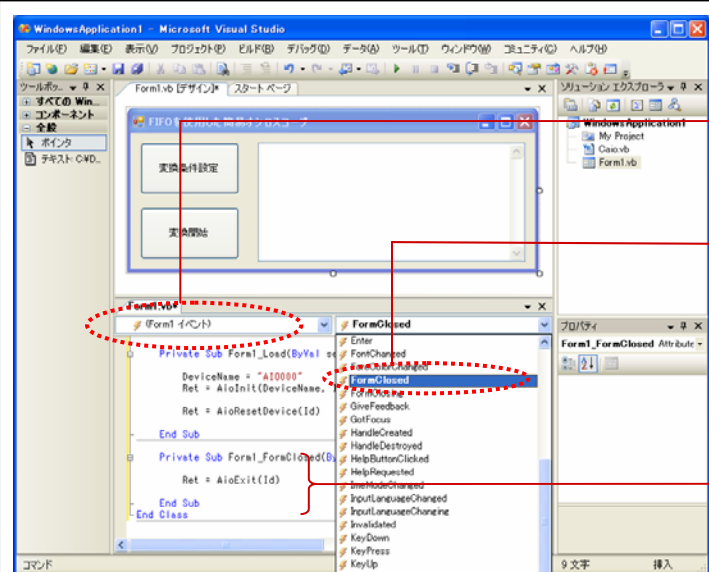
- 引数 Id: AioInIt関数で取得したIDを指定します。

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、割愛しています。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。

4-4-9.プログラム作成手順⑥ (終了処理の追加)

API-AIO (WDM) 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_FormClosedイベント』内で、終了処理関数を実行します。『Form_FormClosedイベント』は『Form』が画面から消去 (クローズド) 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 イベント』を選択します。

② 『プロシージャ』から『FormClosed』を選択します。

③ Private Sub Form1_FormClosed () 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 Form1_FormClosed ()』から『End Sub』の間に記述してください。

```
Ret = AioExit (Id)
```

‘終了処理 (デバイスハンドル開放)

終了処理関数 『AioExit』 リファレンス

■機能 ドライバの終了処理を行います。この関数は、アプリケーションの終了時に実行します。この関数を実行せずにアプリケーションを終了すると、以降デバイスにアクセスできなくなることがあります。

■書式 Visual Basic2005の場合

```
Dim Id As Short  
Dim Ret As Integer  
Ret = AioExit (Id)
```

■引数 Id: 終了するデバイスハンドルを指定します。AioInit関数で取得したIDを指定します。

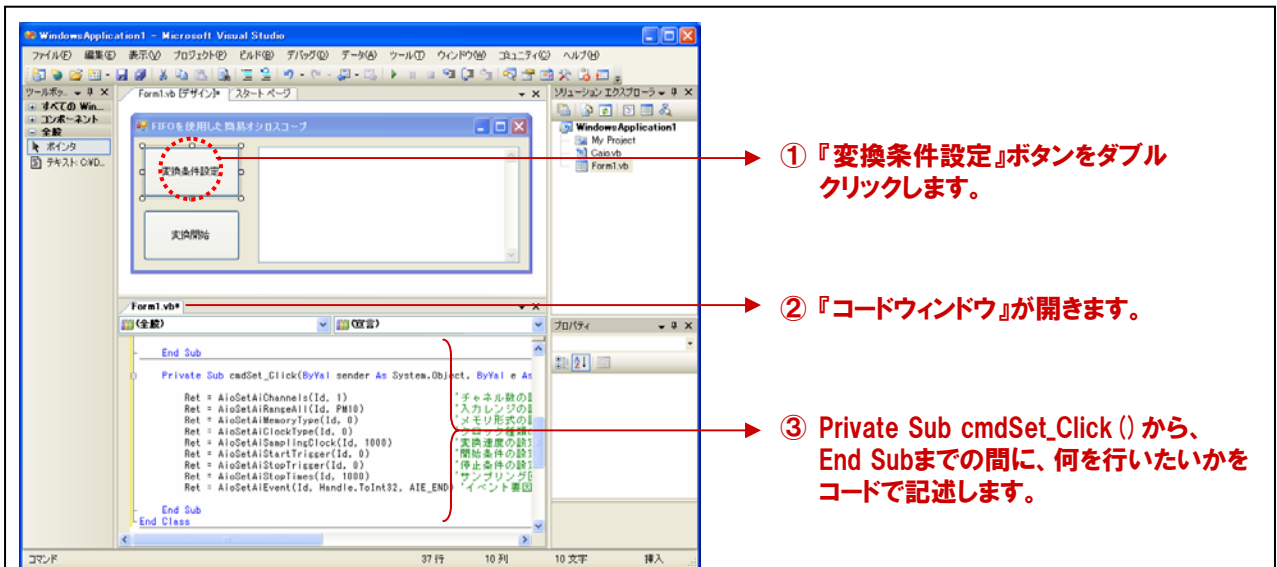
Ret: 終了情報 (戻り値) → 正常終了:0、エラー終了: 0以外 (詳細はヘルプの「戻り値一覧」参照)。

※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しています。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、ドライバの各関数は実行できません。

4-4-10.プログラム作成手順⑦(変換条件設定処理の追加)

メモリ形式 (FIFOまたはRING) や変換クロック、変換速度などアナログ入出力デバイスにセットするアナログ入出力変換条件を設定します。各種設定は、関数を実行するだけで完了です。



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

Ret = AioSetAiChannels (Id, 1)	‘チャンネル数の設定: 1チャンネル
Ret = AioSetAiRangeAll (Id, PM10)	‘入力レンジの設定: ±10VDC
Ret = AioSetAiMemoryType (Id, 0)	‘メモリ形式の設定: FIFO
Ret = AioSetAiClockType (Id, 0)	‘クロック種類の設定: 内部クロック
Ret = AioSetAiSamplingClock (Id, 1000)	‘変換速度の設定: 1000 μ sec
Ret = AioSetAiStartTrigger (Id, 0)	‘開始条件の設定: ソフトウェア
Ret = AioSetAiStopTrigger (Id, 0)	‘停止条件の設定: 設定回数変換終了
Ret = AioSetAiStopTimes (Id, 1000)	‘サンプリング回数の設定: 1000回
Ret = AioSetAiEvent (Id, Handle.ToInt32, AIE_END)	‘イベント要因の設定: デバイス動作終了イベント

解説: 一連の変換条件をデバイスにセットします。関数を実行する順番はとくに決まりはありません。

※: 各関数で設定しているパラメータに関する詳細は、次項からの『関数リファレンス』で解説します。

入力チャンネル数設定処理関数 『AioSetAiChannels』 リファレンス

■機能 変換に使用するアナログ入力チャンネル数の設定を行います。

■書式 Visual Basic2005の場合

```
Dim Id          As Short
Dim Ret         As Integer
Dim AiChannels As Short
Ret = AioSetAiChannels ( Id , AiChannels )
```

■引数 Id: AiOnit関数で取得したIDを指定します。

AiChannels: 変換に使用するチャンネル数を指定します。

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、割愛しています。
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。

全チャンネル入力レンジ設定関数 『AioSetAiRangeAll』 リファレンス

■機能 全チャンネルに対してアナログ入力レンジの設定を行います。

■書式 Visual Basic2005の場合

```
Dim Ret         As Integer
Dim AiRange     As Short
Dim Id          As Short
Ret = AioSetAiRangeAll ( Id , AiRange )
```

■引数 Id : AiOnit 関数で取得したIDを指定します。

AiRange : アナログ入力レンジを以下の範囲からマクロ(標準モジュールファイル内で設定済みの定数)もしくは数値で指定します。設定できる値はデバイスにより異なります。

レンジ	マクロ	値
±10V	PM10	0
±5V	PM5	1
±2.5V	PM25	2
±1.25V	PM125	3
±1V	PM1	4
±0.625V	PM0625	5
±0.5V	PM05	6
±0.3125V	PM03125	7
±0.25V	PM025	8
±0.125V	PM0125	9
±0.1V	PM01	10
±0.05V	PM005	11
±0.025V	PM0025	12
±0.0125V	PM00125	13

レンジ	マクロ	値
0~10V	P10	50
0~5V	P5	51
0~4.095V	P4095	52
0~2.5V	P25	53
0~1.25V	P125	54
0~1V	P1	55
0~0.5V	P05	56
0~0.25V	P025	57
0~0.1V	P01	58
0~0.05V	P005	59
0~0.025V	P0025	60
0~0.0125V	P00125	61
0~20mA	P20MA	100
4~20mA	P4T020MA	101
1~5V	P1T05	150

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、割愛しています。
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 本書にて使用している『ADA16-8/2(CB)L』は、入力レンジは『±10V固定』です。

メモリ形式設定処理関数『AioSetAiMemoryType』リファレンス

■機能 データ格納用メモリ形式の設定を行います。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AiMemoryType As Short
Ret = AioSetAiMemoryType ( Id , AiMemoryType )
```

■引数 Id: AiInIt関数で取得したIDを指定します。

AiMemoryType: 『FIFO = 0』 『RING = 1』。

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、割愛しています。
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 本書では転送方式にデバイスバッファを使用しています。ユーザーバッファでは引数の意味合いが異なります。

クロック種類設定処理関数『AioSetAiClockType』リファレンス

■機能 クロックの種類を取得します。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AiClockType As Short
Ret = AioSetAiClockType ( Id , AiClockType )
```

■引数 Id: AiInIt関数で取得したIDを指定します。

AiClockType: 『内部クロック = 0』 『外部クロック = 1』 『イベントコントローラ = 10』。

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、割愛しています。
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 本書で使用しているADA16-8/2 (CB) Lは、イベントコントローラは使用できません。詳細はヘルプを参照してください。

変換速度設定処理関数『AioSetAiSamplingClock』リファレンス

■機能 内部クロックを使用する場合に、変換速度の設定を行います。内部クロックを使用しない場合には実行する必要はありません。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AiSamplingClock As Single
Ret = AioSetAiSamplingClock ( Id , AiSamplingClock )
```

■引数 Id: AiInIt関数で取得したIDを指定します。

AiSamplingClock: 変換速度を μ sec単位で指定します。デバイスにより設定できる範囲は異なります。

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、割愛しています。
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 本書で使用しているADA16-8/2 (CB) Lは、10~107374182の設定が可能です (初期値:1000)。
詳細はヘルプを参照してください。

変換開始条件設定処理関数『AioSetAiStartTrigger』リファレンス

■機能 変換開始条件の設定を行います。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AiStartTrigger As Short
Ret = AioSetAiStartTrigger ( Id , AiStartTrigger )
```

■引数 Id: AiOnit関数で取得したIDを指定します。

AiStartTrigger: 変換開始条件を以下の範囲から設定します。デバイスにより設定できる値は異なります。

0	ソフトウェア
1	外部トリガ立ち上がり
2	外部トリガ立ち下がり
3	レベル比較
4	インレンジ比較
5	アウトレンジ比較
10	イベントコントローラ出力

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、割愛しています。
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 本書で使用しているADA16-8/2 (CB) Lは、AiStartTriggerは0、1、2、3の設定が可能です。

変換停止条件設定処理関数『AioSetAiStopTrigger』リファレンス

■機能 クロックの種類を取得します。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AiStopTrigger As Short
Ret = AioSetAiStopTrigger ( Id , AiStopTrigger )
```

■引数 Id: AiOnit関数で取得したIDを指定します。

AiStopTrigger: 変換停止条件を以下の範囲から設定します。デバイスにより設定できる値は異なります。

0	設定回数変換終了
1	外部トリガ立ち上がり
2	外部トリガ立ち下がり
3	レベル比較
4	コマンド(AioStopAi)
5	インレンジ比較
6	アウトレンジ比較
10	イベントコントローラ出力

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、割愛しています。
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 本書で使用しているADA16-8/2 (CB) Lは、AiStopTriggerは0、1、2、3、4の設定が可能です。

サンプリング回数設定処理関数『AioSetAiStopTimes』リファレンス

■機能 サンプリング回数の設定を行います。この設定は、AioSetAiStopTrigger関数で変換停止条件を設定回数変換終了に設定した場合に必要になります。変換停止条件が設定回数変換終了以外の場合には実行する必要はありません。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AiStopTimes As Integer
Ret = AioSetAiStopTimes ( Id , AiStopTimes )
```

■引数 Id: AiOnit関数で取得したIDを指定します。

AiStopTimes: サンプリング回数を指定します。

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、割愛しています。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 本書で使用しているADA16-8/2 (CB) Lの設定可能なAiStopTimesは1~4294967295です。

イベント処理処理関数『AioSetAiEvent』リファレンス

■機能 アナログ入力に関するWindowメッセージ通知のイベント要因を設定します。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim hWnd As Integer
Dim AiEvent As Integer
Ret = AioSetAiEvent ( Id , hWnd , AiEvent )
```

■引数 Id: AiOnit関数で取得したIDを指定します。

hWnd: Windowハンドルを指定します。

AiEvent: イベント要因を以下の範囲からマクロもしくは数値で指定します。AiEventはビット単位で以下のような意味を持ち、これらを組み合わせて指定可能です。デバイスバッファ使用時とユーザーバッファ使用時で使用可能なイベント要因が異なります。

イベント要因	デバイスバッファ	ユーザーバッファ	マクロ	値
AD変換開始条件成立イベント	○	○	AIE_START	0000002H
リビート終了イベント	○	○	AIE_RPTEND	0000010H
デバイス動作終了イベント	○	○	AIE_END	0000020H
指定サンプリング回数格納イベント	○	×	AIE_DATA_NUM	0000080H
指定転送数毎イベント	×	○	AIE_DATA_TSF	0000100H
オーバーフローイベント	○	○	AIE_OFERR	0001000H
サンプリングクロックエラーイベント	○	○	AIE_SCERR	0002000H
AD変換エラーイベント	○	○	AIE_ADERR	0004000H

この関数で設定されたイベント要因は、イベントメッセージルーチンにメッセージとして通知されます。メッセージの種類は、[4-4-6.③]を参照してください。

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、割愛しています。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 本書で使用しているADA16-8/2 (CB) Lでは、デバイスバッファ形式を使用しています。

4-4-11.プログラム作成手順⑧(メモリ領域のリセットと変換開始命令の追加)

デバイスのバッファメモリ領域のクリア(リセット)と変換動作開始を行います。【4-4-10.】で設定した条件を元に
変換動作を行い変換が終了するとデバイスは割り込みを発信、WndProc関数が呼ばれ処理を実行します。

① 『変換開始』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。

```
Ret = AioResetAiMemory (Id)           'メモリリセット
Ret = AioStartAi (Id)                 '変換開始
```

メモリリセット処理関数 『AioResetAiMemory』 リファレンス

■機能 デバイスメモリ、またはドライバメモリをリセットします。この関数はAioSetAiTransferMode関数で変換データ転送方式をデバイスバッファモードに設定した場合のみ使用できます。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Ret = AioResetAiMemory ( Id )
```

■引数 Id: Aiolnit関数で取得した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、割愛しています。
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。

A/D変換開始処理関数 『AioStartAi』 リファレンス

■機能 設定された条件に基づいてA/D変換を開始します。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Ret = AioStartAi ( Id )
```

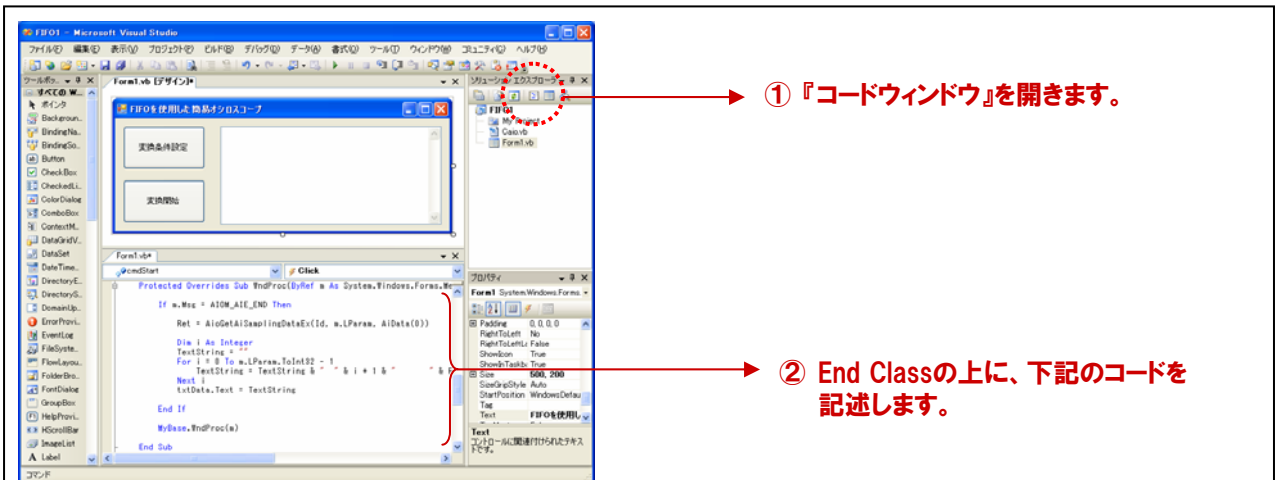
■引数 Id: Aiolnit関数で取得したIDを指定します。

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、割愛しています。
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。

4-4-12.プログラム作成手順⑨ (イベント発生時処理 (データ取得処理) の追加)

設定した条件で変換動作が終了すると、デバイスは割り込み信号を発信し、デバイスドライバがそれを受け取り、アプリケーションへ決められたメッセージを発信します。そのメッセージに対してWndProc関数が呼ばれ、処理を行います。ここでは、データ取得と表示の処理を行います。



```
Protected Overrides Sub WndProc (ByRef m As System.Windows.Forms.Message)
If m.Msg = AIOM_AIE_END Then

    Ret = AioGetAiSamplingDataEx (Id, m.LParam, AiData (0)) '変換データ取得
    Dim i As Integer '変換データ表示
    TextString = ""
    For i = 0 To m.LParam.ToInt32 - 1
        TextString = TextString & " " & i + 1 & " " & Format (AiData (i), "0.000") & vbCrLf
    Next i
    txtData.Text = TextString

End If
MyBase.WndProc(m)
End Sub
```

解説: イベントの引数『Message』には、変換が正常に終了した場合、先の変換条件設定で設定したメッセージが渡されてきます。その判断にIf~End If構文を使用しています。イベントの引数『lParam』には、【4-4-6.③】で説明したとおり、今回は『サンプリングの回数:1000』が渡されてきます。その情報をもとに、データ取得関数を実行し、テキストボックスへの表示を行っています。下記のリファレンスでは、サンプリング数を固定値として説明しています。

データ取得関数 (電圧/電流値) 『AioGetAiSamplingDataEx』 リファレンス

機能 デバイスメモリ (ドライバメモリ) から指定サンプリング分のデータを読み込みます。変換データは電圧または電流で格納されます。この関数はAioSetAiTransferMode関数で変換データ転送方式をデバイスバッファモードに設定した場合のみ使用できます。変換データ転送方式がユーザーバッファの場合には使用できません。

書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AiSamplingTimes As Integer
Dim AiData As Single
Ret = AioGetAiSamplingDataEx ( Id, AiSamplingTimes, AiData (0) )
```

引数 Id: AiIoInit関数で取得したIDを指定します。

AiSamplingTimes: 格納するサンプリング数を格納した変数を指定します。

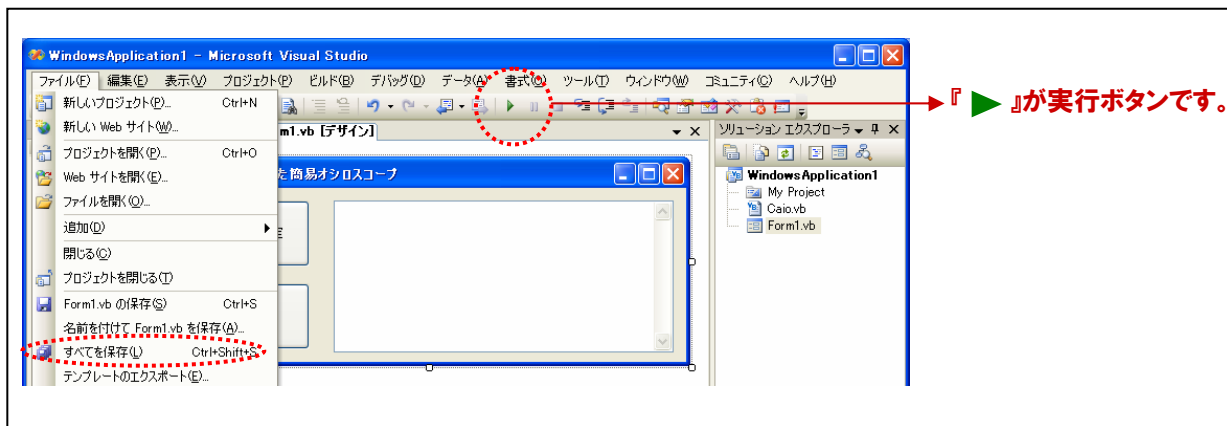
AiData: 変換データを格納する配列変数の先頭を指定します。変換データは電圧または電流で格納されます。

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、割愛しています。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。

4-4-13.プログラムの実行

- ① 『ファイル』メニューの中から『すべてを保存』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。保存が終わりましたら、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。

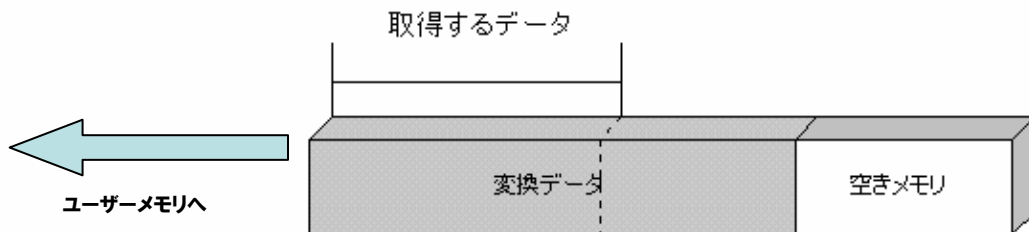


- ② 入力チャンネル0には、ファンクションジェネレータがBNCケーブルによって接続されています。ファンクションジェネレータの設定は、出力電圧=±10VDC、出力波形=サイン波です。画面上の『変換条件設定ボタン』、『変換開始ボタン』の順で実行してください。1000 μsec間隔のデータが1000個入力されます。



デバイスバッファメモリ (カード内のメモリ) から、パソコン上のメモリへの転送イメージ

カード内のタイマカウンタ (内部クロック) の周期でサンプリングされたデータは、デバイスバッファ (カード内のメモリ) へ蓄積されていきます。本プログラムの場合、1000回サンプリングを終えた時点で、割り込み機能を使用して、変換終了をアプリケーションに通知し、そのタイミングでデータをパソコン上のメモリへ転送して、各種処理・解析などを行います。



4-4-14. FIFOメモリを使用した高速サンプリング (簡易オシロスコープ : データ表示) リスト

Dim Ret	As Integer	‘戻り値用変数
Dim Id	As Short	‘ID格納用変数
Dim AiData (999)	As Single	‘変換データ格納用変数 (1000個分の配列変数)
Dim DeviceName	As String	‘デバイス名設定用変数
Dim TextString	As String	‘データ表示用変数

Private Sub Form1_Load (ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

DeviceName = "AI0000"	‘デバイス名を変数に格納
Ret = AioInit (DeviceName, Id)	‘初期化処理 (デバイスハンドル取得)
Ret = AioResetDevice (Id)	‘指定デバイスのリセット

End Sub

Private Sub Form1_FormClosed (ByVal sender As Object, ByVal e As System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed

Ret = AioExit (Id)	‘終了処理 (デバイスハンドル開放)
--------------------	--------------------

End Sub

Private Sub cmdSet_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdSet.Click

Ret = AioSetAiChannels (Id, 1)	‘チャンネル数の設定: 1チャンネル
Ret = AioSetAiRangeAll (Id, PM10)	‘入力レンジの設定: ±10VDC
Ret = AioSetAiMemoryType (Id, 0)	‘メモリ形式の設定: FIFO
Ret = AioSetAiClockType (Id, 0)	‘クロック種類の設定: 内部クロック
Ret = AioSetAiSamplingClock (Id, 1000)	‘変換速度の設定: 1000 μ sec
Ret = AioSetAiStartTrigger (Id, 0)	‘開始条件の設定: ソフトウェア
Ret = AioSetAiStopTrigger (Id, 0)	‘停止条件の設定: 設定回数変換終了
Ret = AioSetAiStopTimes (Id, 1000)	‘サンプリング回数の設定: 1000回
Ret = AioSetAiEvent (Id, Handle.ToInt32, AIE_END)	‘イベント要因の設定: デバイス動作終了イベント

End Sub

Private Sub cmdStart_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdStart.Click

Ret = AioResetAiMemory (Id)	‘メモリリセット
Ret = AioStartAi (Id)	‘変換開始

End Sub

Protected Overrides Sub WndProc (ByRef m As System.Windows.Forms.Message)

If m.Msg = AIOM_AIE_END Then

Ret = AioGetAiSamplingDataEx (Id, m.LParam, AiData (0)) ‘変換データ取得

Dim i As Integer ‘変換データ表示

TextString = ""

For i = 0 To m.LParam.ToInt32 - 1

TextString = TextString & " " & i + 1 & " " & Format (AiData (i), "0.000") & vbCrLf

Next i

txtData.Text = TextString

End If

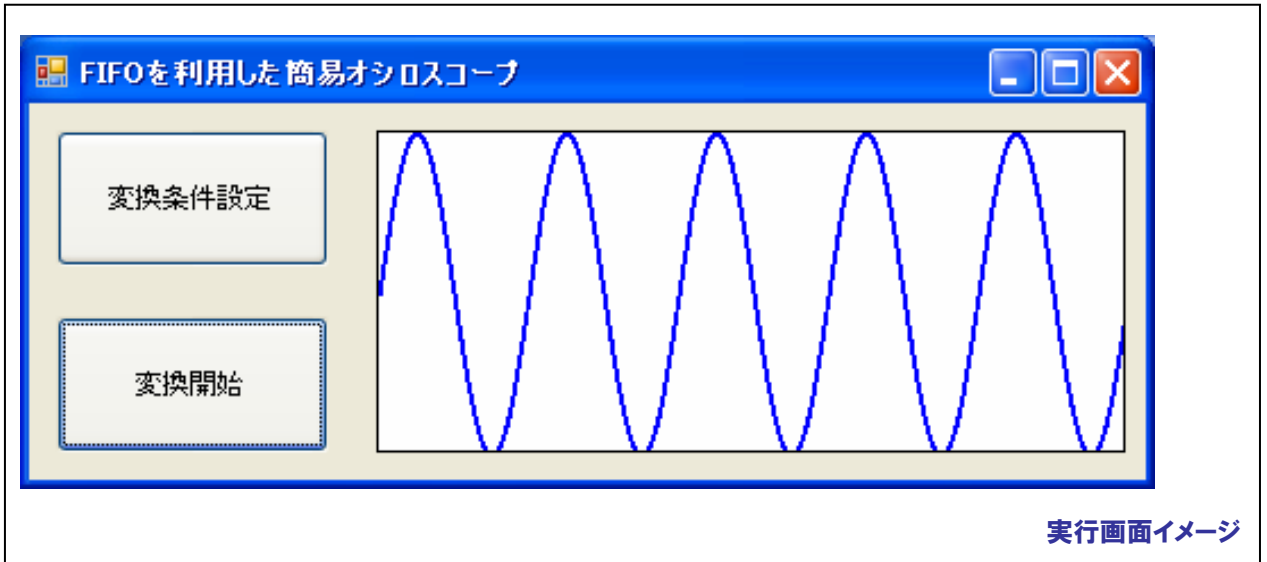
End Sub

4-5.FIFOメモリを使用した高速サンプリング (簡易オシロスコープ：波形表示)

【4-4.】では、内部クロックとデバイスに搭載されているバッファメモリを使用した高速サンプリングプログラム (簡易オシロスコープ) を作成しました。【4-4.】で作成したプログラムでは、取得データの表示機能のみでしたが、ここでは、取得データの波形表示 (グラフ表示) プログラムを作成します。なお、一部の使用関数を除き、アナログ入力処理は【4-4.】と共通です。

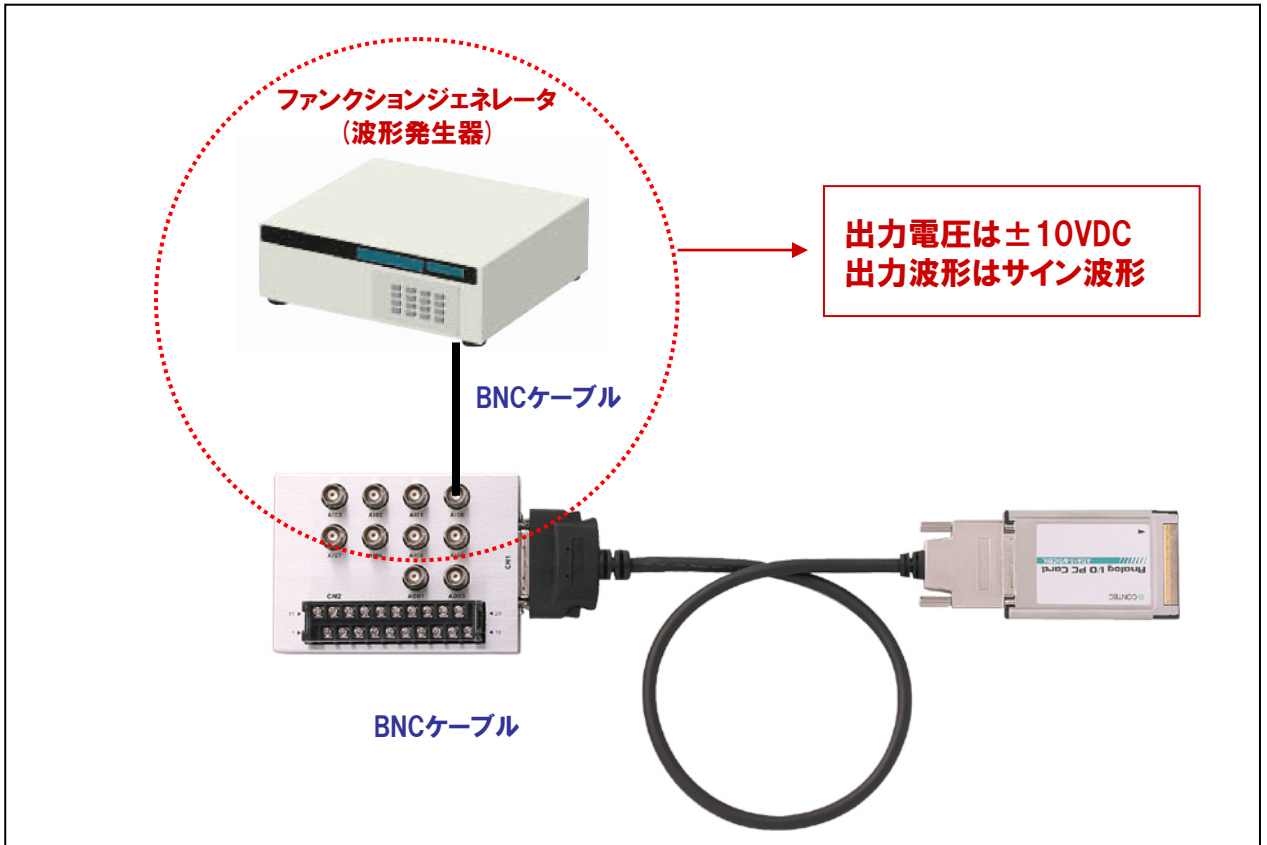
4-5-1.プログラム概要

バッファメモリ (FIFO形式) を使用して、1000 μ sec周期のデータを1000回サンプリングし、ピクチャボックスに波形表示するプログラムです。アナログ入出力カード『ADA16-8/2 (CB) L』の『アナログ入力0ch』のみ使用します。



4-5-2.実行環境

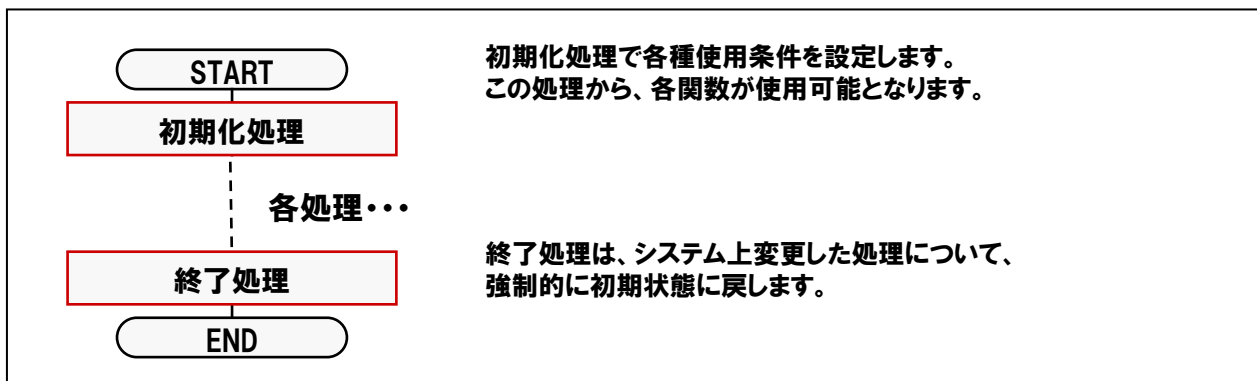
【3-6-3.実習環境の構築】の環境を使用します。ファンクションジェネレータは、次の設定を事前に行っています。出力アナログ信号は ± 10 VDC (VDC:直流を示す)、出力波形はサイン波としています。



4-5-3.プログラム フローチャート

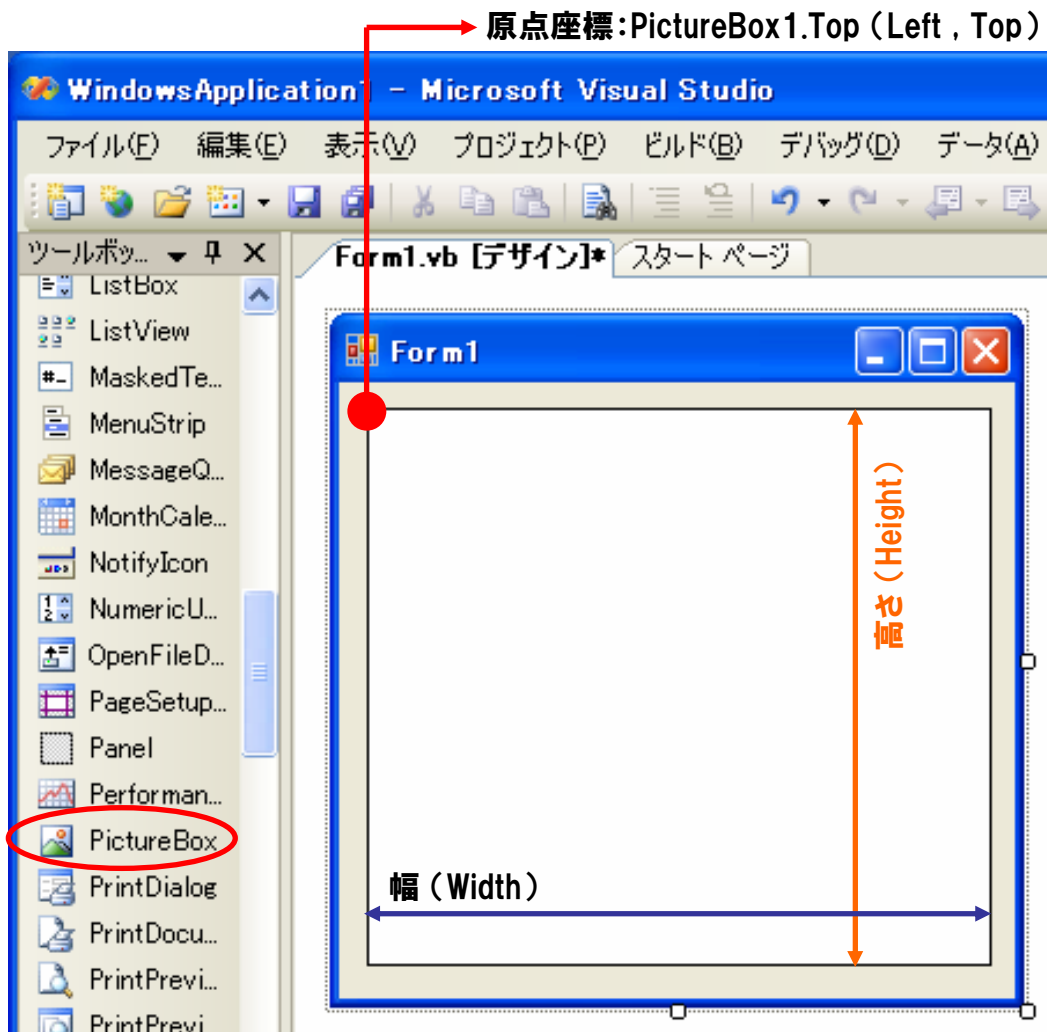


※: API-AIO (WDM) の処理体系は、初期化処理ではじまり、終了処理で終了する決まりがあります。初期化処理および、終了処理の専用関数を用意していますので、その関数を実行します。



4-5-4.参考資料Visual Basic2005のピクチャーボックスコントロールの基礎知識

Visual Basic2005の標準コントロールである、ピクチャーボックスは、ビットマップ、アイコン、メタファイルといった画像を表示できるほか、『描画メソッド』を用いてグラフィックが描画可能なコントロールです。



Visual Basic2005の内部座標の単位は「Pixel」を使用しています。
1センチ = 40Pixel、1インチ = 101.6Pixel

PictureBox上で描画を行うためには、最初に以下の書式に従って変数の定義を行い、PictureBoxを描画可能な状態にする必要があります。

Dim 変数名 As Graphics = オブジェクト名.CreateGraphics

また、描画終了時には終了処理を行います。

変数名.Dispose ()

◎座標 (X1, Y1) と (X2, Y2) 間に直線を引く場合の書式は、

変数名. DrawLine (<ペンの指定>, X1, Y1, X2, Y2)

4-5-5.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)

[4-2-2.]~[4-2-7.]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
◎Text = 『FIFOを使用した簡易オシロスコープ』

■ 変換条件設定ボタン
◎Text = 『変換条件設定』
◎Name = 『cmdSet』

■ PictureBoxコントロール
◎Name = 『Graph』

■ 変換開始ボタン
◎Text = 『変換開始』
◎Name = 『cmdStart』

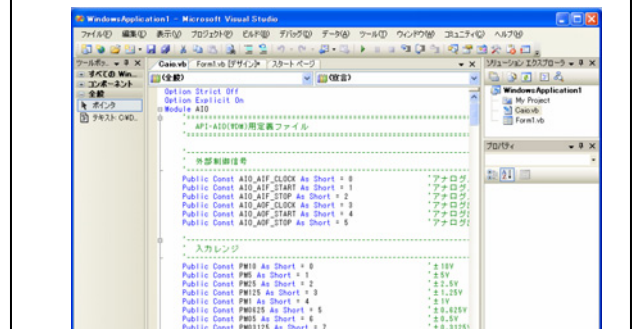
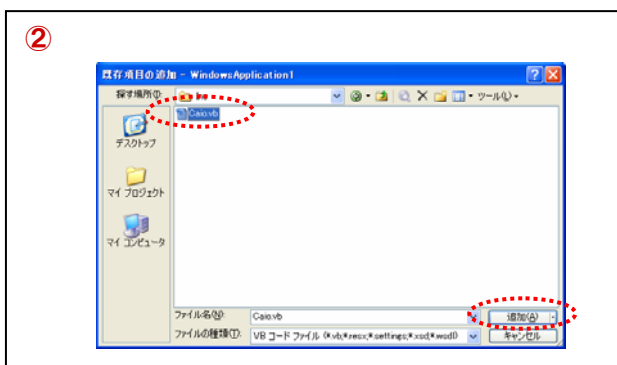
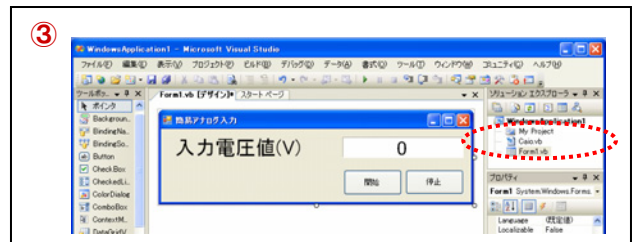
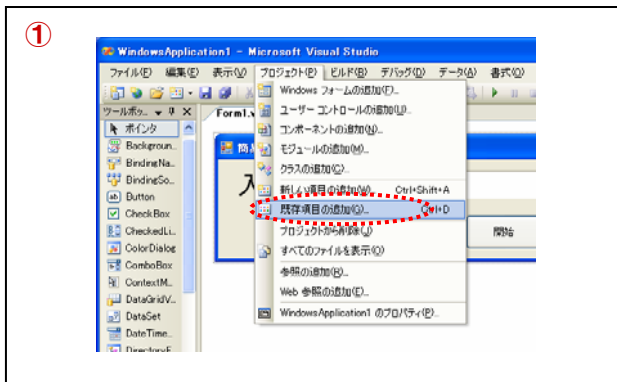
4-5-6.プログラム作成手順② (標準モジュールファイルの追加)

Visual BasicでAPI-AIO (WDM) 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 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 (Caio.vb) 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) ¥AIOWDM¥Sample¥Inc¥Caio.vb

- ③ 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。



Caio.vb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、定数の宣言などがされているのが分かります。

4-5-7.プログラム作成手順③ (割り込み処理の実現:WndProc)

API-AIO (WDM) で割り込みを実現させるためにWndProc関数を使用します。割り込みのソフト的な処理説明に関しては、【4-4-6】を参照してください。WndProc関数は、Windowsメッセージと内容を取得する関数です。

Visual Basic2005、Visual C#で使用し、API-AIO (WDM) では以下のメッセージを扱います。

アナログ入力メッセージ要因	マクロ	値
AD変換開始条件成立イベント	AIOM_AIE_START	1000H
リピート終了イベント	AIOM_AIE_RPTEND	1001H
デバイス動作終了イベント	AIOM_AIE_END	1002H
指定サンプリング回数格納イベント	AIOM_AIE_DATA_NUM	1003H
指定転送数毎イベント	AIOM_AIE_DATA_TSF	1007H
オーバーフローイベント	AIOM_AIE_OFERR	1004H
サンプリングクロックエラーイベント	AIOM_AIE_SCERR	1005H
A/D変換エラーイベント	AIOM_AIE_ADERR	1006H
アナログ出力メッセージ要因	マクロ	値
DA変換開始条件成立イベント	AIOM_AOE_START	1020H
リピート終了イベント	AIOM_AOE_RPTEND	1021H
デバイス動作終了イベント	AIOM_AOE_END	1022H
指定転送数毎イベント	AIOM_AOE_DATA_TSF	1027H
指定サンプリング回数出力イベント	AIOM_AOE_DATA_NUM	1023H
サンプリングクロックエラーイベント	AIOM_AOE_SCERR	1025H
D/A変換エラーイベント	AIOM_AOE_ADERR	1026H
カウンタメッセージ要因	マクロ	値
動作終了イベント	AIOM_CNTE_END	1040H
動作開始条件成立イベント	AIOM_CNTE_START	1041H
比較カウント一致イベント	AIOM_CNTE_DATA_NUM	1042H
カウントオーバーランイベント	AIOM_CNTE_ORERR	1043H
タイマメッセージ要因	マクロ	値
インターバル成立イベント	AIOM_TME_INT	1060H

Visual Basic2005でのイベントメッセージルーチンは次の書式になります。

`Protected Overrides Sub WndProc (ByRef m As System.Windows.Forms.Message`

◎m.Msg : メッセージ番号が渡されます。

◎m.WParam : 下位2バイトにIDが渡されます。上位2バイトは現在使用しません。

◎m.LParam : イベントごとに固有のパラメータが渡されます。本書では『デバイス動作終了イベント』を使用しますので、下表のとおり『m.LParam』引数には『現在のサンプリング回数』が渡されてきます。

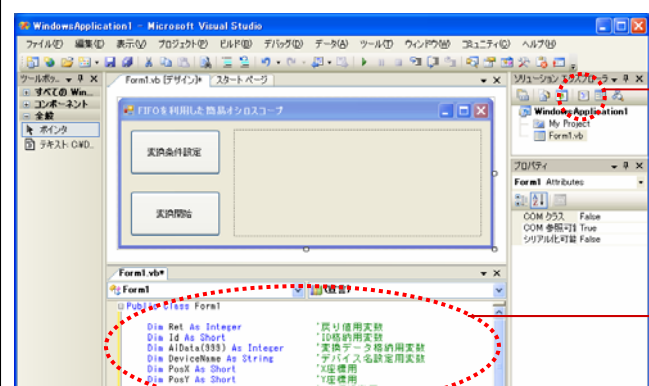
イベント要因	パラメータ
AD変換開始条件成立イベント	なし
リピート終了イベント	現在のリピート回数
デバイス動作終了イベント	現在のサンプリング回数
指定サンプリング回数出力イベント	現在のサンプリング回数
指定転送数毎イベント	現在の転送回数
オーバーフローエラーイベント	現在のサンプリング回数
サンプリングクロックエラーイベント	現在のサンプリング回数
AD変換エラーイベント	現在のサンプリング回数

4-5-8.プログラム作成手順④ (変数の追加)

簡易オシロスコープ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』 - 『プログラム』 - 『CONTEC API-PAC (W32)』 - 『AIOWDM』 - 『API-AIO (WDM) 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。

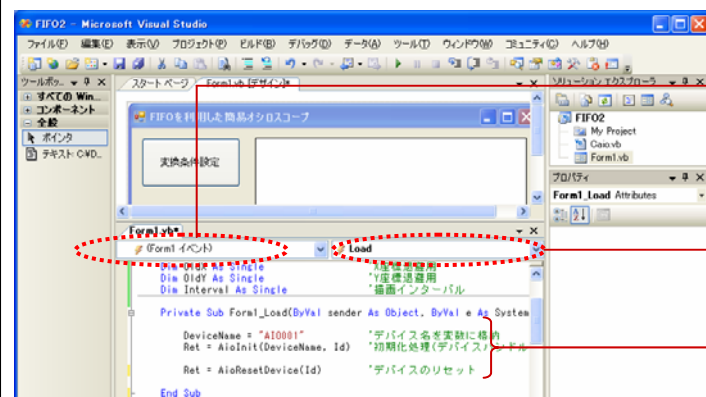
② Public Class Form1からEnd Classの間に下記の変数を記述します。

Dim Ret	As Integer	‘戻り値用変数
Dim Id	As Short	‘ID格納用変数
Dim AiData (999)	As Integer	‘変換データ格納用変数
Dim DeviceName	As String	‘デバイス名設定用変数
Dim PosX	As Single	‘X座標用
Dim PosY	As Single	‘Y座標用
Dim OldX	As Single	‘X座標退避用
Dim OldY	As Single	‘Y座標退避用
Dim Interval	As Single	‘描画インターバル

4-5-9.プログラム作成手順⑤ (初期化処理とアナログ入力デバイスリセットの追加)

API-AIO (WDM) 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、変数を初期化の際に使われます。

また、アナログ入力デバイスのリセットを行います。



① コードウィンドウを開き、『オブジェクト』から『Form1 イベント』を選択します。

② 『プロセス』から『Load』を選択します。

③ Private Sub Form1_Load () から、End Subの間に初期化処理関数とデバイスリセット関数を記述します。

下記のコードを『Private Sub Form1_Load ()』から『End Sub』の間に記述してください。

DeviceName = "AIO000"	‘デバイス名を変数に格納
Ret = AioInit (DeviceName, Id)	‘初期化処理 (デバイスハンドル取得)
Ret = AioResetDevice (Id)	‘デバイスのリセット

初期化処理関数『AioInIt』リファレンス

- 機能 デバイスファイルを作成し、以降デバイスを使用可能にします。デバイスにアクセスするには、まずこの関数を実行する必要があります。この関数がプロセスで初めて実行された時には、内部のパラメータがすべてデフォルト値に設定されます。デバイス内にレジスタを持つ場合はそのレジスタもデフォルト値に設定されます。AioExit関数を実行せずに続けてAioInItを実行しても、内部のパラメータはデフォルトに戻りません。内部パラメータをデフォルト値に戻すには、AioResetDeviceを使用します。

- 書式 Visual Basic2005の場合

```
Dim Ret As Integer
Dim DeviceName As String
Dim Id As Short
Ret = AioInIt ( DeviceName , Id )
```

- 引数 DeviceName: デバイスマネージャのプロパティページ、または設定ツールで設定したデバイス名を指定します。
Id: ID (デバイスハンドル)を受け取る変数を指定します。以降の関数は、この変数に格納された値を用いてアクセスできます。

デバイス名に関して



3-5.手順 ④:ドライバソフトウェアの初期設定の項目で設定した、デバイス名を使用します。

本書では、デフォルト値『AI0000』を使用します。

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、割愛しています。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。

デバイスリセット処理関数『AioResetDevice』リファレンス

- 機能 デバイスのリセット、ドライバの初期化を行います。

- 書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Ret = AioResetDevice ( Id )
```

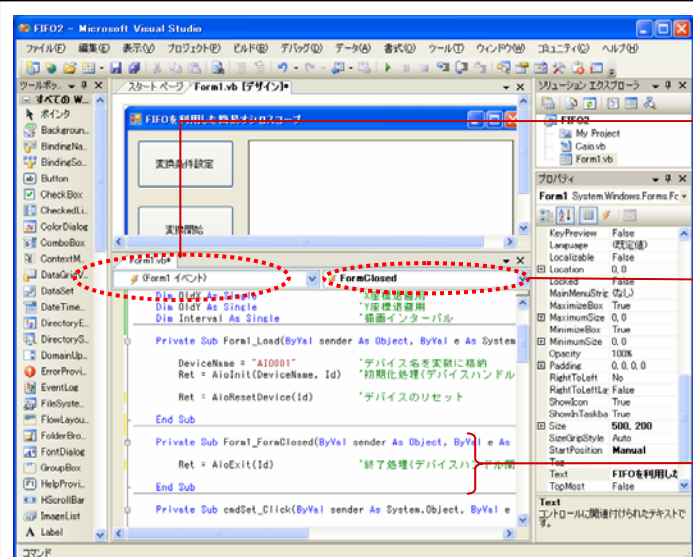
- 引数 Id: AioInIt関数で取得したIDを指定します。

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、割愛しています。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。

4-5-10.プログラム作成手順⑥ (終了処理の追加)

API-AIO (WDM) 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_FormClosedイベント』内で、終了処理関数を実行します。『Form_FormClosedイベント』は『Form』が画面から消去 (クローズド) 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『Form1 イベント』を選択します。

② 『プロセス』から『FormClosed』を選択します。

③ Private Sub Form1_FormClosed () 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_FormClosed ()』から『End Sub』の間に記述してください。

```
Ret = AioExit (Id)
```

‘終了処理 (デバイスハンドル開放)

終了処理関数 『AioExit』 リファレンス

■機能 ドライバの終了処理を行います。この関数は、アプリケーションの終了時に実行します。この関数を実行せずにアプリケーションを終了すると、以降デバイスにアクセスできなくなることがあります。

■書式 Visual Basic2005の場合

```
Dim Id As Short  
Dim Ret As Integer  
Ret = AioExit (Id)
```

■引数 Id: 終了するデバイスハンドルを指定します。AioInit関数で取得したIDを指定します。

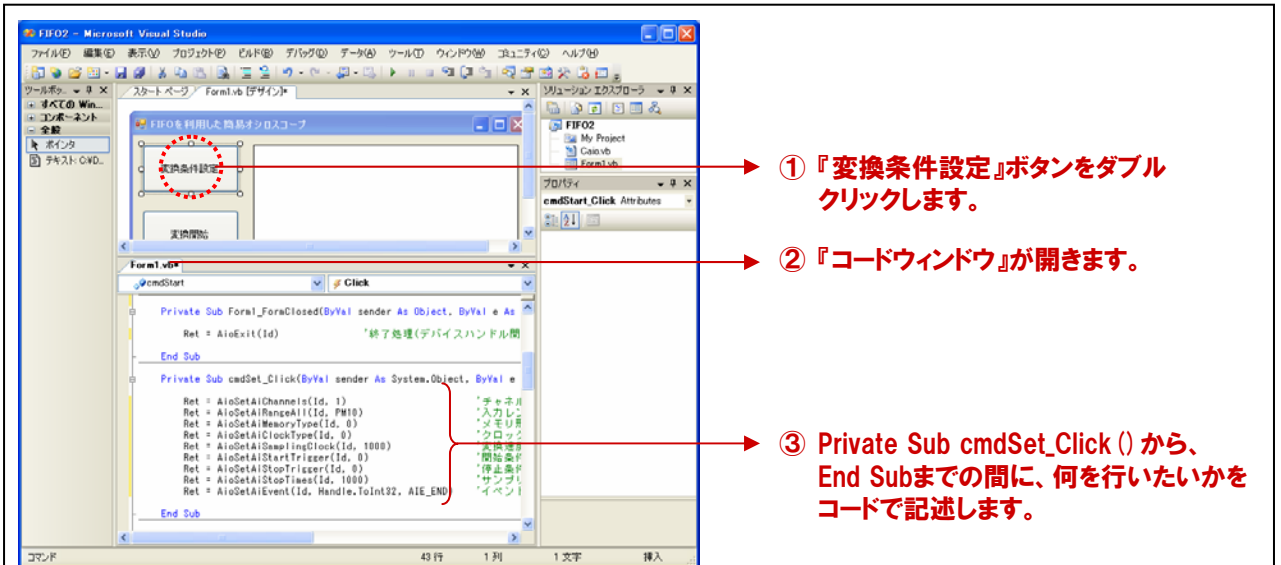
Ret: 終了情報 (戻り値) → 正常終了:0、エラー終了: 0以外 (詳細はヘルプの「戻り値一覧」参照)。

※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しています。
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、ドライバの各関数は実行できません。

4-5-11.プログラム作成手順⑦(変換条件設定処理の追加)

メモリ形式 (FIFOまたはRING) や変換クロック、変換速度などアナログ入出力デバイスにセットするアナログ入出力変換条件を設定します。各種設定は、関数を実行するだけで完了です。



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

Ret = AioSetAiChannels (Id, 1)	'チャンネル数の設定:1チャンネル
Ret = AioSetAiRangeAll (Id, PM10)	'入力レンジの設定:±10VDC
Ret = AioSetAiMemoryType (Id, 0)	'メモリ形式の設定:FIFO
Ret = AioSetAiClockType (Id, 0)	'クロック種類の設定:内部クロック
Ret = AioSetAiSamplingClock (Id, 1000)	'変換速度の設定:1000 μ sec
Ret = AioSetAiStartTrigger (Id, 0)	'開始条件の設定:ソフトウェア
Ret = AioSetAiStopTrigger (Id, 0)	'停止条件の設定:設定回数変換終了
Ret = AioSetAiStopTimes (Id, 1000)	'サンプリング回数の設定:1000回
Ret = AioSetAiEvent (Id, Handle.ToInt32, AIE_END)	'イベント要因の設定:デバイス動作終了イベント

解説: 一連の変換条件をデバイスにセットします。関数を実行する順番はとくに決まりはありません。

※:各関数で設定しているパラメータに関する詳細は、次項からの『関数リファレンス』を参照ください。

入力チャンネル数設定処理関数『AioSetAiChannels』リファレンス

■機能 変換に使用するアナログ入力チャンネル数の設定を行います。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AiChannels As Short
Ret = AioSetAiChannels ( Id , AiChannels )
```

■引数 Id: AiOnit関数で取得したIDを指定します。

AiChannels: 変換に使用するチャンネル数を指定します。

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、割愛しています。
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。

全チャンネル入力レンジ設定関数『AioSetAiRangeAll』リファレンス

■機能 全チャンネルに対してアナログ入力レンジの設定を行います。

■書式 Visual Basic2005の場合

```
Dim Ret As Integer
Dim Id As Short
Dim AiRange As Short
Ret = AioSetAiRangeAll ( Id , AiRange )
```

■引数 Id : AiOnit 関数で取得したIDを指定します。

AiRange : アナログ入力レンジを以下の範囲からマクロ(標準モジュールファイル内で設定済みの定数)もしくは数値で指定します。設定できる値はデバイスにより異なります。

レンジ	マクロ	値
±10V	PM10	0
±5V	PM5	1
±2.5V	PM25	2
±1.25V	PM125	3
±1V	PM1	4
±0.625V	PM0625	5
±0.5V	PM05	6
±0.3125V	PM03125	7
±0.25V	PM025	8
±0.125V	PM0125	9
±0.1V	PM01	10
±0.05V	PM005	11
±0.025V	PM0025	12
±0.0125V	PM00125	13

レンジ	マクロ	値
0~10V	P10	50
0~5V	P5	51
0~4.095V	P4095	52
0~2.5V	P25	53
0~1.25V	P125	54
0~1V	P1	55
0~0.5V	P05	56
0~0.25V	P025	57
0~0.1V	P01	58
0~0.05V	P005	59
0~0.025V	P0025	60
0~0.0125V	P00125	61
0~20mA	P20MA	100
4~20mA	P4T020MA	101
1~5V	P1T05	150

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、割愛しています。
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 本書にて使用している『ADA16-8/2(CB)L』は、入力レンジは『±10V固定』です。

メモリ形式設定処理関数『AioSetAiMemoryType』リファレンス

■機能 データ格納用メモリ形式の設定を行います。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AiMemoryType As Short
Ret = AioSetAiMemoryType ( Id , AiMemoryType )
```

■引数 Id: AiInIt関数で取得したIDを指定します。

AiMemoryType: 『FIFO = 0』 『RING = 1』。

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、割愛しています。
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 本書では転送方式にデバイスバッファを使用しています。ユーザーバッファでは引数の意味合いが異なります。

クロック種類設定処理関数『AioSetAiClockType』リファレンス

■機能 クロックの種類を取得します。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AiClockType As Short
Ret = AioSetAiClockType ( Id , AiClockType )
```

■引数 Id: AiInIt関数で取得したIDを指定します。

AiClockType: 『内部クロック = 0』 『外部クロック = 1』 『イベントコントローラ = 10』。

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、割愛しています。
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 本書で使用しているADA16-8/2 (CB) Lは、イベントコントローラは使用できません。詳細はヘルプを参照してください。

変換速度設定処理関数『AioSetAiSamplingClock』リファレンス

■機能 内部クロックを使用する場合に、変換速度の設定を行います。内部クロックを使用しない場合には実行する必要はありません。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AiSamplingClock As Single
Ret = AioSetAiSamplingClock ( Id , AiSamplingClock )
```

■引数 Id: AiInIt関数で取得したIDを指定します。

AiSamplingClock: 変換速度を μ sec単位で指定します。デバイスにより設定できる範囲は異なります。

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、割愛しています。
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 本書で使用しているADA16-8/2 (CB) Lは、10~107374182の設定が可能です (初期値:1000)。
詳細はヘルプを参照してください。

変換開始条件設定処理関数『AioSetAiStartTrigger』リファレンス

■機能 変換開始条件の設定を行います。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AiStartTrigger As Short
Ret = AioSetAiStartTrigger ( Id , AiStartTrigger )
```

■引数 Id: AiOnit関数で取得したIDを指定します。

AiStartTrigger: 変換開始条件を以下の範囲から設定します。デバイスにより設定できる値は異なります。

0	ソフトウェア
1	外部トリガ立ち上がり
2	外部トリガ立ち下がり
3	レベル比較
4	インレンジ比較
5	アウトレンジ比較
10	イベントコントローラ出力

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、割愛しています。
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 本書で使用しているADA16-8/2 (CB) Lは、AiStartTriggerは0、1、2、3の設定が可能です。

変換停止条件設定処理関数『AioSetAiStopTrigger』リファレンス

■機能 クロックの種類を取得します。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AiStopTrigger As Short
Ret = AioSetAiStopTrigger ( Id , AiStopTrigger )
```

■引数 Id: AiOnit関数で取得したIDを指定します。

AiStopTrigger: 変換停止条件を以下の範囲から設定します。デバイスにより設定できる値は異なります。

0	設定回数変換終了
1	外部トリガ立ち上がり
2	外部トリガ立ち下がり
3	レベル比較
4	コマンド(AioStopAi)
5	インレンジ比較
6	アウトレンジ比較
10	イベントコントローラ出力

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、割愛しています。
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 本書で使用しているADA16-8/2 (CB) Lは、AiStopTriggerは0、1、2、3、4の設定が可能です。

サンプリング回数設定処理関数『AioSetAiStopTimes』リファレンス

■機能 サンプリング回数の設定を行います。この設定は、AioSetAiStopTrigger関数で変換停止条件を設定回数変換終了に設定した場合に必要になります。変換停止条件が設定回数変換終了以外の場合には実行する必要はありません。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AiStopTimes As Integer
Ret = AioSetAiStopTimes ( Id , AiStopTimes )
```

■引数 Id: AiOnit関数で取得したIDを指定します。

AiStopTimes: サンプリング回数を指定します。

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、割愛しています。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 本書で使用しているADA16-8/2 (CB) Lの設定可能なAiStopTimesは1~4294967295です。

イベント処理処理関数『AioSetAiEvent』リファレンス

■機能 アナログ入力に関するWindowメッセージ通知のイベント要因を設定します。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim hWnd As Integer
Dim AiEvent As Integer
Ret = AioSetAiEvent ( Id , hWnd , AiEvent )
```

■引数 Id: AiOnit関数で取得したIDを指定します。

hWnd: Windowハンドルを指定します。

AiEvent: イベント要因を以下の範囲からマクロもしくは数値で指定します。AiEventはビット単位で以下のような意味を持ち、これらを組み合わせて指定可能です。デバイスバッファ使用時とユーザーバッファ使用時で使用可能なイベント要因が異なります。

イベント要因	デバイスバッファ	ユーザーバッファ	マクロ	値
AD変換開始条件成立イベント	○	○	AIE_START	0000002H
リビート終了イベント	○	○	AIE_RPTEND	0000010H
デバイス動作終了イベント	○	○	AIE_END	0000020H
指定サンプリング回数格納イベント	○	×	AIE_DATA_NUM	0000080H
指定転送数毎イベント	×	○	AIE_DATA_TSF	0000100H
オーバーフローイベント	○	○	AIE_OFERR	0001000H
サンプリングクロックエラーイベント	○	○	AIE_SCERR	0002000H
AD変換エラーイベント	○	○	AIE_ADERR	0004000H

この関数で設定されたイベント要因は、イベントメッセージルーチンにメッセージとして通知されます。メッセージの種類は、[4-4-6.③]を参照してください。

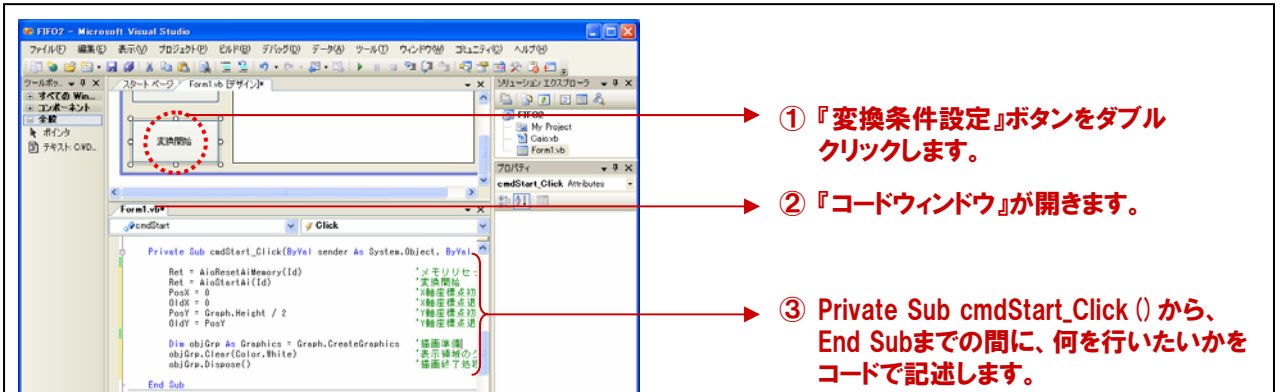
Ret: 終了情報 (戻り値) → 正常終了:0、エラー終了: 0以外 (詳細はヘルプの「戻り値一覧」参照)。

※ 本書では、戻り値の確認(エラー処理)は、誌面の都合上、割愛しています。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 本書で使用しているADA16-8/2 (CB) Lでは、デバイスバッファ形式を使用しています。

4-5-12.プログラム作成手順⑧(メモリ領域のリセットと変換開始、グラフ初期化処理の追加)

デバイスのバッファメモリ領域のクリア(リセット)と変換動作開始を行います。【4-4-10.】で設定した条件を元に
変換動作を行い、変換が終了するとデバイスは割り込みを発生、WndProc関数が呼ばれ処理を実行します。



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。

```
Ret = AioResetAiMemory (Id)          'メモリリセ
Ret = AioStartAi (Id)                '変換開始
PosX = 0                             'X軸座標点初期化
OldX = 0                             'X軸座標点退避用
PosY = Graph.ScaleHeight / 2        'Y軸座標点初期化
OldY = PosY                          'Y軸座標点退避
Dim objGrp As Graphics = Graph.CreateGraphics 'グラフ描画準備
objGrp.Clear (Color.White)          '表示領域のクリア
objGrp.Dispose ()                   '描画終了処理
```

メモリリセット処理関数 『AioResetAiMemory』 リファレンス

■機能 デバイスメモリ、またはドライバメモリをリセットします。この関数はAioSetAiTransferMode関数で変換データ転送方式をデバイスバッファモードに設定した場合のみ使用できます。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Ret = AioResetAiMemory ( Id )
```

■引数 Id: Aiolnit関数で取得した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、割愛しています。
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。

A/D変換開始処理関数 『AioStartAi』 リファレンス

■機能 設定された条件に基づいてA/D変換を開始します。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Ret = AioStartAi ( Id )
```

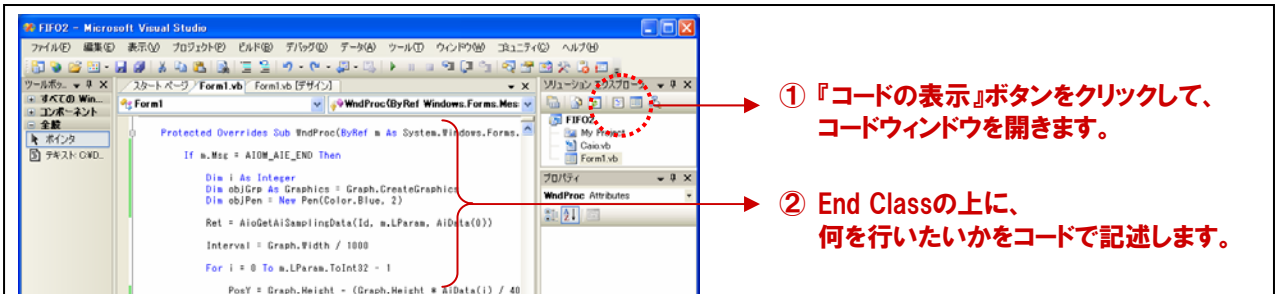
■引数 Id: Aiolnit関数で取得したIDを指定します。

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、割愛しています。
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。

4-5-13.プログラム作成手順⑨ (イベント発生時処理 (データ取得処理) の追加)

設定した条件で変換動作が終了すると、デバイスは割り込み信号を発信し、デバイスドライバがそれを受け取り、アプリケーションへ決められたメッセージを発信します。そのメッセージに対してWndProc関数が呼ばれ、処理を行います。ここでは、データ取得と表示の処理を行います。



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 End Classの上に、何を行いたいかをコードで記述します。

End Classの上に、下記のコードを記述します。

```
Protected Overrides Sub WndProc (ByRef m As System.Windows.Forms.Message)
    If m.Msg = AIOM_AIE_END Then
        Ret = AioGetAiSamplingData (Id, m.LParam, AiData (0)) '変換データ取得 (バイナリ)

        Interval = Graph.Width / 1000 '描画インターバルの指定
        Dim objGrp As Graphics = Graph.CreateGraphics '描画準備
        Dim objPen = New Pen (Color.Blue, 2) '描画用ペンの定義
        Dim i As Integer 'ループ用変数

        For i = 0 To m.LParam.ToInt32 - 1 'グラフ描画処理
            PosY = Graph.Height - Graph.Height * AiData (i) / 65536
            If PosY = Graph.Height Then
                PosY = Graph.Height - 1
            End If
            objGrp.DrawLine (objPen, OldX, OldY, PosX, PosY)
            OldY = PosY
            OldX = PosX
            PosX = OldX + Interval
        Next i
        objGrp.Dispose () '描画終了処理
        objPen.Dispose ()
    End If
    MyBase.WndProc (m) 'ベースクラスのWndProc処理
End Sub
```

解説: イベントの引数『m.Msg』には、変換が正常に終了した場合、先の変換条件設定で設定したメッセージが渡されてきます。その判断にIf~End If構文を使用しています。イベントの引数『lParam』には、【4-4-6.③】で説明したとおり、今回は『サンプリングの回数:1000』が渡されてきます。その情報をもとに、データ取得関数を実行し、ピクチャーボックスへの波形表示を行っています。For~Next、If~End If構文はVisual Basic2005の説明書を参照ください。下記のリファレンスでは、サンプリング数を固定値として説明しています。

データ取得関数 (バイナリ値) 『AioGetAiSamplingData』 リファレンス

■機能 デバイスメモリ (ドライバメモリ) から指定サンプリング分のデータを読み込みます。変換データはバイナリ値で格納されます。この関数はAioSetAiTransferMode関数で変換データ転送方式をデバイスバッファモードに設定した場合のみ使用できます。変換データ転送方式がユーザーバッファの場合には使用できません。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AiSamplingTimes As Integer
Dim AiData As Integer
Ret = AioGetAiSamplingData ( Id , AiSamplingTimes , AiData (0) )
```

■引数 Id: AiIoInit関数で取得したIDを指定します。

AiSamplingTimes: 格納するサンプリング数を格納した変数を指定します。

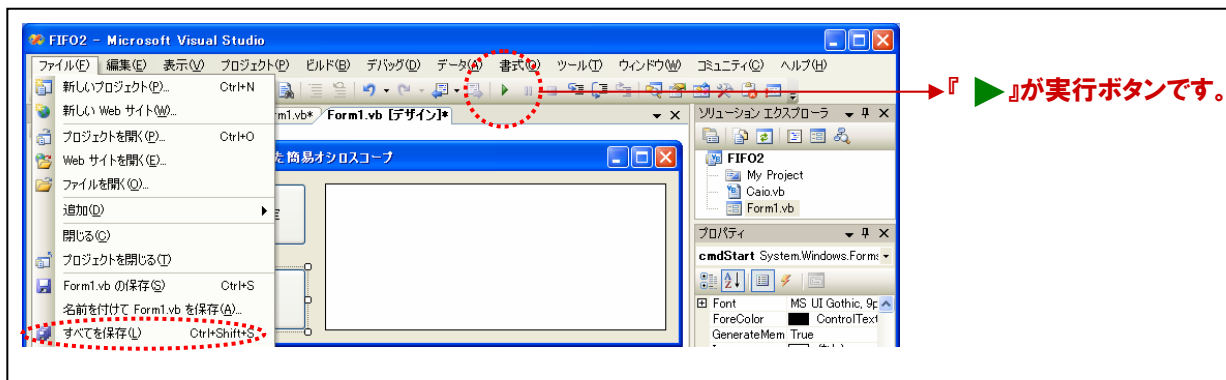
AiData: 変換データを格納する配列変数の先頭を指定します。変換データはバイナリ値で格納されます。

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)。

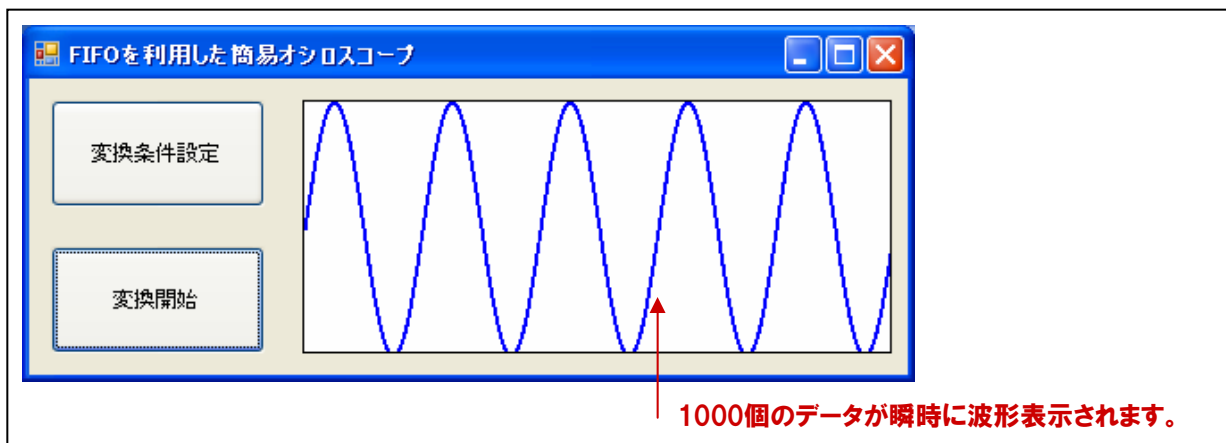
※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しています。実システムにおいては、関数を実行した後にエラー処理のコードを記述します。エラー処理の方法は、各サンプルプログラムを参照してください。

4-5-14.プログラムの実行

- ① 『ファイル』メニューの中から『すべてを保存』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。保存が終わりましたら、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 入力チャンネル0には、ファンクションジェネレータがBNCケーブルによって接続されています。ファンクションジェネレータの設定は、出力電圧=±10VDC、出力波形=サイン波です。画面上の『変換条件設定ボタン』、『変換開始ボタン』の順で実行してください。1000 μsec間隔のデータが1000個入力され波形表示されます。



TOPICS:『もっと、手軽に簡単にプログラムを組みたい方へ...』

コンテックでは、計測プログラムをもっと手軽に簡単に、しかも高機能に作成するための開発ツール、計測システム開発用ActiveXコンポーネント集『ACX-PAC (W32)』を発売しています。

データのグラフ表示はもちろん、ボード/カードを簡単に制御できるコントロール、スイッチやランプといった画面表示用コントロール、FFT解析が行える解析用コントロールなどが満載です。詳細は、第5章『ActiveXによるコンポーネントプログラミング』、および『付録』を参照ください。

なお、弊社ホームページの『計測システム開発用ActiveXコンポーネント集 ACX-PAC (W32) Ver.4.11』のスペシャルサイトでは、無償体験版CD-ROMのご請求ができます。是非、一度ご参照ください。

計測システム開発用 Windows Vista/XP/2000/NT4.0/Me/98/95対応

Active Xコンポーネント集

体験版CD-ROM無料配布中! ACX-PAC(W32) Ver.4.11
New Version

ハード制御/計算/解析/表示パーツ満載の
オールインパッケージ

●200種類以上のアドオンボード/PCカード/
USBデバイスに対応

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

4-5-15. FIFOメモリを使用した高速サンプリング (簡易オシロスコープ : 波形表示) リスト

Dim Ret	As Integer	'戻り値用変数
Dim Id	As Short	'ID格納用変数
Dim AiData (999)	As Integer	'変換データ格納用変数
Dim DeviceName	As String	'デバイス名設定用変数
Dim PosX	As Single	'X座標用
Dim PosY	As Single	'Y座標用
Dim OldX	As Single	'X座標退避用
Dim OldY	As Single	'Y座標退避用
Dim Interval	As Single	'描画インターバル

Private Sub Form1_Load (ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

```
DeviceName = "AIO000"           'デバイス名を変数に格納
Ret = AioInit (DeviceName, Id)  '初期化处理 (デバイスハンドル取得)

Ret = AioResetDevice (Id)       'デバイスのリセット
```

End Sub

Private Sub Form1_FormClosed (ByVal sender As Object, ByVal e As System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed

```
Ret = AioExit (Id)              '終了処理 (デバイスハンドル開放)
```

End Sub

Private Sub cmdSet_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdSet.Click

```
Ret = AioSetAiChannels (Id, 1)   'チャンネル数の設定: 1チャンネル
Ret = AioSetAiRangeAll (Id, PM10) '入力レンジの設定: ±10VDC
Ret = AioSetAiMemoryType (Id, 0) 'メモリ形式の設定: FIFO
Ret = AioSetAiClockType (Id, 0)  'クロック種類の設定: 内部クロック
Ret = AioSetAiSamplingClock (Id, 1000) '変換速度の設定: 1000 μ sec
Ret = AioSetAiStartTrigger (Id, 0) '開始条件の設定: ソフトウェア
Ret = AioSetAiStopTrigger (Id, 0) '停止条件の設定: 設定回数変換終了
Ret = AioSetAiStopTimes (Id, 1000) 'サンプリング回数の設定: 1000回
Ret = AioSetAiEvent (Id, CMessage1.Window, AIE_END) 'イベント要因の設定: デバイス動作終了イベント
```

End Sub

次頁に続く →

```
Private Sub cmdStart_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
cmdStart.Click
```

```
Ret = AioResetAiMemory (Id)           'メモリリセット
Ret = AioStartAi (Id)                 '変換開始

PosX = 0                              'X軸座標点初期化
OldX = 0                              'X軸座標点退避用
PosY = Graph.ScaleHeight / 2         'Y軸座標点初期化
OldY = PosY                           'Y軸座標点退避
Dim objGrp As Graphics = Graph.CreateGraphics '描画準備
objGrp.Clear (Color.White)           '表示領域のクリア
objGrp.Dispose ()                   '描画終了処理
```

```
End Sub
```

```
Protected Overrides Sub WndProc (ByRef m As System.Windows.Forms.Message)
```

```
If m.Msg = AIOM_AIE_END Then

    Ret = AioGetAiSamplingData (Id, m.LParam, AiData (0)) '変換データ取得 (バイナリ)

    Interval = Graph.Width / 1000      '描画インターバルの指定
    Dim objGrp As Graphics = Graph.CreateGraphics '描画準備
    Dim objPen = New Pen (Color.Blue, 2) '描画用ペンの定義
    Dim i As Integer                    'ループ用変数

    For i = 0 To m.LParam.ToInt32 - 1  'グラフ描画処理
        PosY = Graph.Height - Graph.Height * AiData (i) / 65536

        If PosY = Graph.Height Then
            PosY = Graph.Height - 1
        End If

        objGrp.DrawLine (objPen, OldX, OldY, PosX, PosY)
        OldY = PosY
        OldX = PosX
        PosX = OldX + Interval
    Next i

    objGrp.Dispose ()                 '描画終了処理
    objPen.Dispose ()

End If

MyBase.WndProc (m)                  'ベースクラスのWndProc処理
```

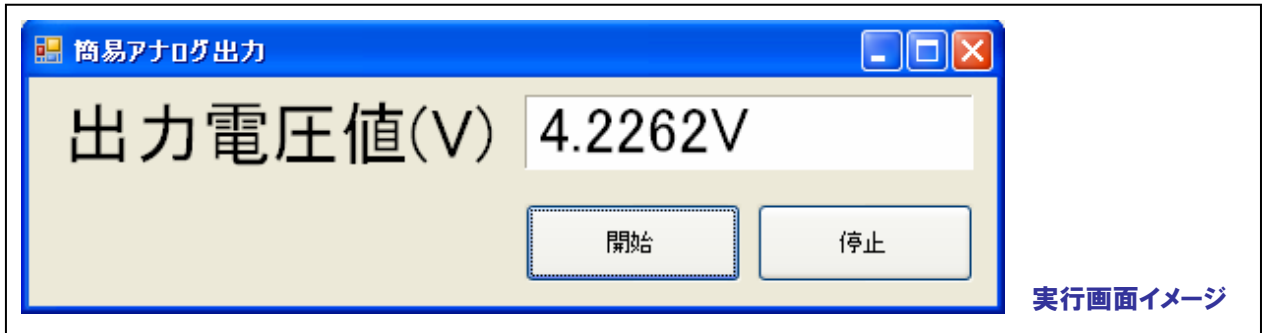
```
End Sub
```


4-6. タイマコントロールによる簡易連続アナログ出力プログラム

【4-3.】～【4-5.】はアナログ入力、すなわち、外部からのアナログ信号をデジタル信号に変換してパソコンに入力、そのデータの表示や波形での表示を行うプログラムを作成してきました。
ここからは、アナログ出力、すなわちパソコンからのデジタル信号をアナログ信号に変換して、外部出力を行うプログラムを作成していきます。最初に作成するプログラムは変換開始トリガおよび変換クロックは『ソフトウェア』です。

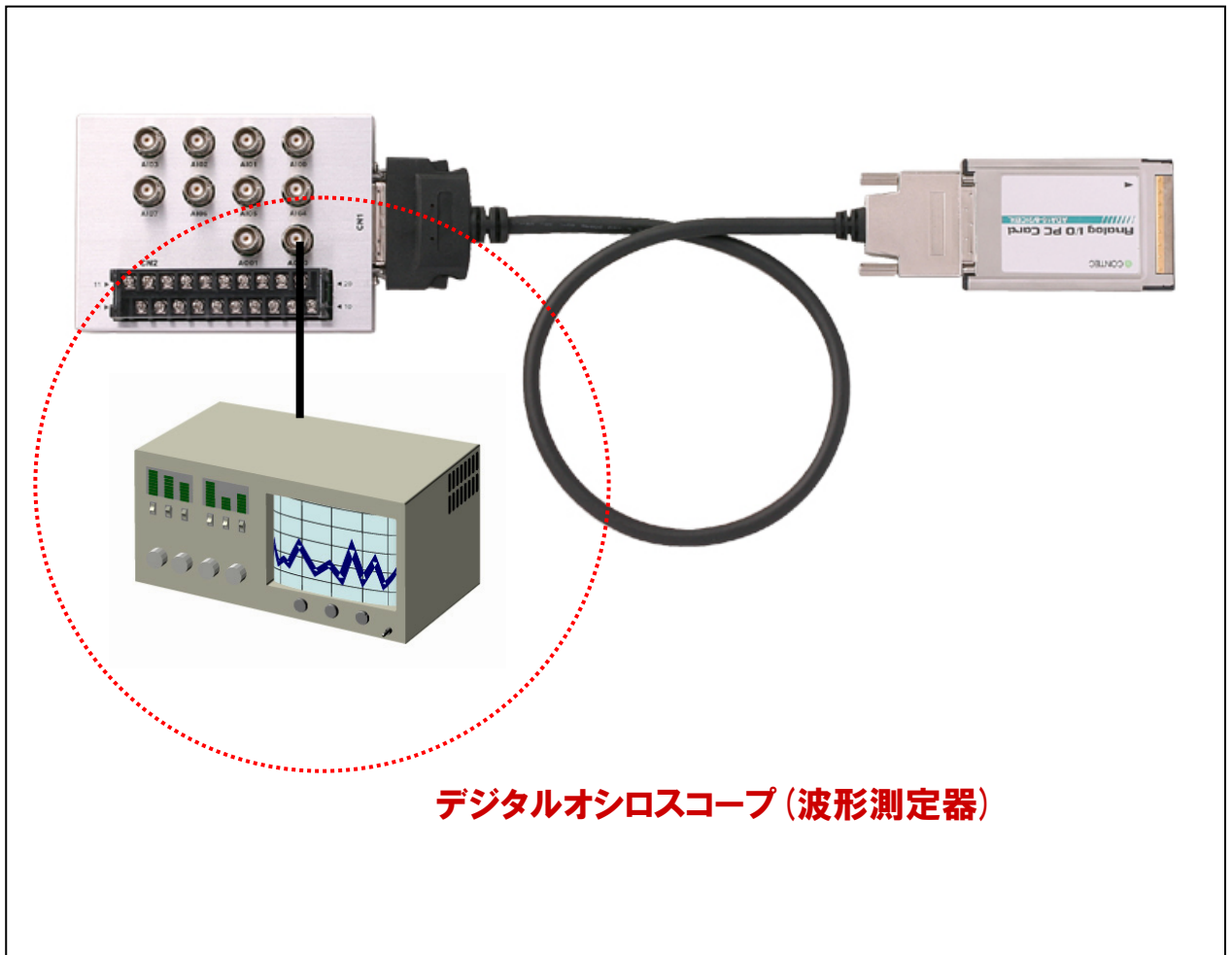
4-6-1. プログラム概要

タイマコントロールの周期 (100msec) でアナログ入出力カード『ADA16-8/2 (CB) L』の『アナログ出力0ch』からサイン波形データをデジタルオシロスコープへ出力します。画面で出力データ確認も行います。

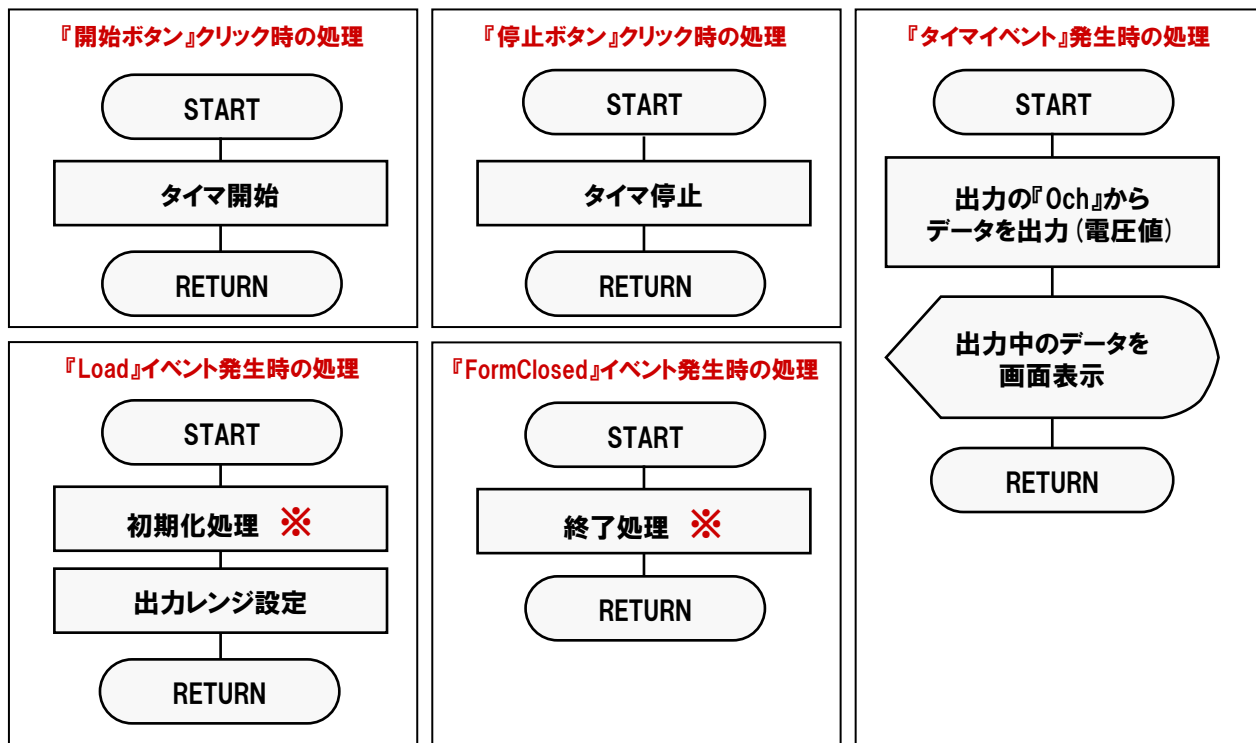


4-6-2. 実行環境

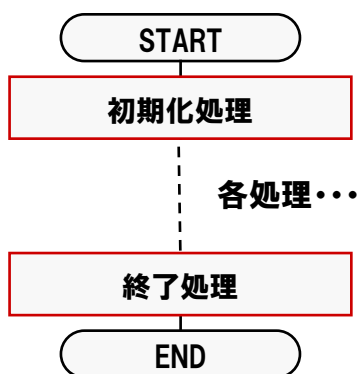
【3-6-3. 実習環境の構築】の環境を使用します。使用するデジタルオシロスコープの操作方法に関しては、それぞれの説明書に従ってください。



4-6-3.プログラム フローチャート



※： API-AIO (WDM) の処理体系は、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、その関数を実行します。



初期化処理で各種使用条件を設定します。
この処理から、各関数が使用可能となります。

終了処理は、システム上変更した処理について、
強制的に初期状態に戻します。

TOPICS

『API-AIO (WDM) について』

『API-AIO (WDM)』は、従来のアナログ入出力ドライバ『API-AIO (98/PC) **』と比較して、「より使いやすく便利に」、「より高機能に」を目指した、新アナログ入出力用ドライバです。ユーザーインターフェースの改善をはじめ、ハードウェアの機能をフルに活用することにより、アナログ入出力ボード/カードを使用したアプリケーション開発をより強力にサポートします。

- シンプルで使いやすい機能ごとに分類された関数を提供。
- 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。
- 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。

注1) 『API-AIO (WDM)』と従来ドライバ『API-AIO (98/PC) **』では
互換性がありません。

注2) 使用するボード/カードによっては、『API-AIO (WDM)』のみの
サポートとなります。

その他の違いは、API-AIO (WDM) のヘルプファイルを参照してください。
ヘルプファイルには、これらの情報の他に 親切・丁寧な関数リファレンス、
チュートリアルなどが収録されています。



『スタート』 - 『プログラム』 - 『CONTEC API-PAC (W32)』 - 『AIOWDM』 - 『API-AIO (WDM) HELP』

4-6-4.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)

[4-2-2.]~[4-2-7.]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 Studio window with a Windows Form titled 'Form1.vb' containing several controls. Red dashed boxes and arrows point to specific controls and their properties in the Properties window:

- Form**: Text = 『簡易アナログ出力』
- Labelコントロール**: Name = 『IblData』
- 開始ボタン**: Text = 『開始』, Name = 『cmdStart』
- 停止ボタン**: Text = 『停止』, Name = 『cmdStop』
- タイマコントロール**: Name = 『tmrTimer』, Enabled = 『False』, Interval = 『100』
- Labelコントロール**: Name = 『IblTitle』, Text = 『出力電圧値 (V)』

4-6-5.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-AIO (WDM) 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 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 (Caio.vb) 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) ¥AIOWDM¥Sample¥Inc¥Caio.vb

③ 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。

The screenshots show the following steps:

- Step 1:** The 'Project' menu is open, and 'Add Standard Module' (標準モジュールの追加) is selected.
- Step 2:** The 'Add Existing Item' dialog box is shown with 'Caio.vb' selected in the file list.
- Step 3:** The 'Caio.vb' file is added to the project, and the code editor shows the module's contents, including API declarations and constants.

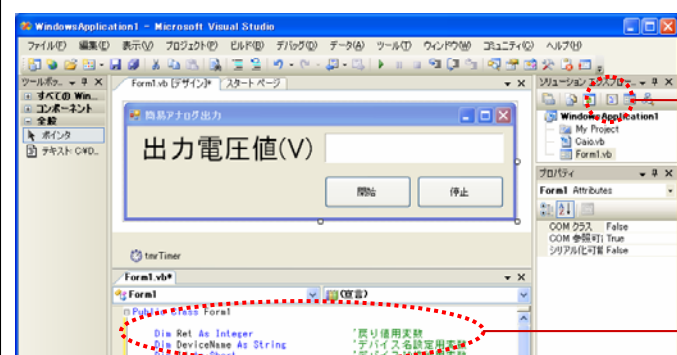
Caio.vbをダブルクリックで開くと、ドライバソフトウェアが提供している関数の情報(参照先)や、構造体の宣言、定数の宣言などがされているのが分かります。

4-6-6.プログラム作成手順③ (変数の追加)

簡易アナログ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』 - 『プログラム』 - 『CONTEC API-PAC (W32)』 - 『AIOWDM』 - 『API-AIO (WDM) HELP』



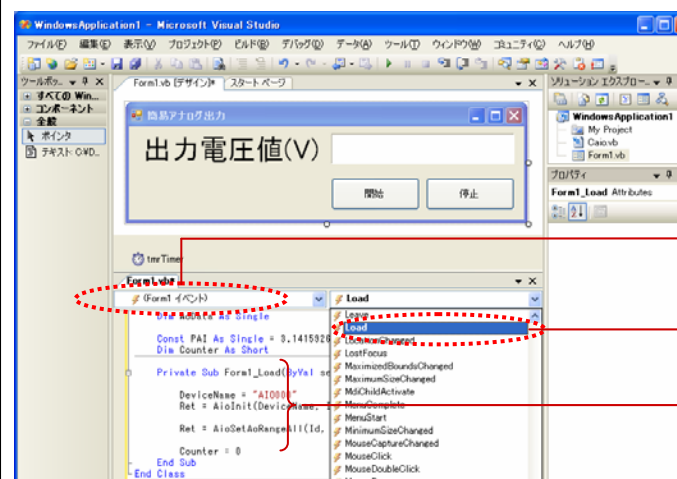
① 『コードの表示』ボタンをクリックして、コードウィンドウを開きます。

② Public Class Form1から、End Classの間に下記の変数を記述します。

Dim Ret	As Integer	‘戻り値用変数
Dim DeviceName	As String	‘デバイス名設定用変数
Dim Id	As Integer	‘デバイスID格納用変数
Dim AoData	As Single	‘出力電圧値格納用変数
Dim Counter	As Integer	‘カウンタ (サイン波形算出用) 変数

4-6-7.プログラム作成手順④ (初期化処理とアナログ出力初期設定、変数初期化の追加)

API-AIO (WDM) 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、変数を初期化の際に使われます。また、アナログ出力に関する初期設定 (本プログラムでは、出力レンジの設定のみ) と変数の初期化処理を行います。



① コードウィンドウを開き、『オブジェクト』から『Form1 イベント』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と入力レンジ設定関数を記述します。

下記のコードを『Private Sub Form1_Load ()』から『End Sub』の間に記述してください。

DeviceName = "AIO000"	‘デバイス名を変数に格納
Ret = AioInIt (DeviceName, Id)	‘初期化処理
Ret = AioSetAoRangeAll (Id, PM10)	‘出力レンジ設定 (±10VDC)
Counter = 0	‘サイン波算出用変数0クリア

※:各関数で設定しているパラメータに関する詳細は、次項からの『関数リファレンス』で解説します。

初期化処理関数『AioInIt』リファレンス

■機能 デバイスファイルを作成し、以降デバイスを使用可能にします。デバイスにアクセスするには、まずこの関数を実行する必要があります。この関数がプロセスで初めて実行された時には、内部のパラメータがすべてデフォルト値に設定されます。デバイス内にレジスタを持つ場合はそのレジスタもデフォルト値に設定されます。AioExit関数を実行せずに続けてAioInItを実行しても、内部のパラメータはデフォルトに戻りません。内部パラメータをデフォルト値に戻すには、AioResetDeviceを使用します。

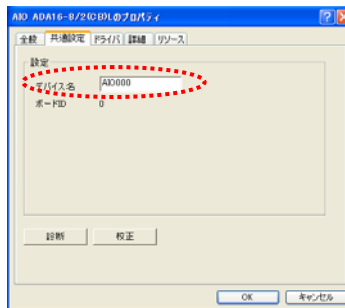
■書式 Visual Basic2005の場合

```
Dim Ret As Integer  
Dim DeviceName As String  
Dim Id As Short  
Ret = AioInIt ( DeviceName , Id )
```

■引数 DeviceName: デバイスマネージャのプロパティページ、または設定ツールで設定したデバイス名を指定します。

Id: ID (デバイスハンドル) を受け取る変数を指定します。以降の関数は、この変数に格納された値を用いてアクセスできます。

デバイス名に関して



3-5.手順 ④:ドライバソフトウェアの初期設定の項目で設定した、デバイス名を使用します。

本書では、デフォルト値『AI0000』を使用します。

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)。

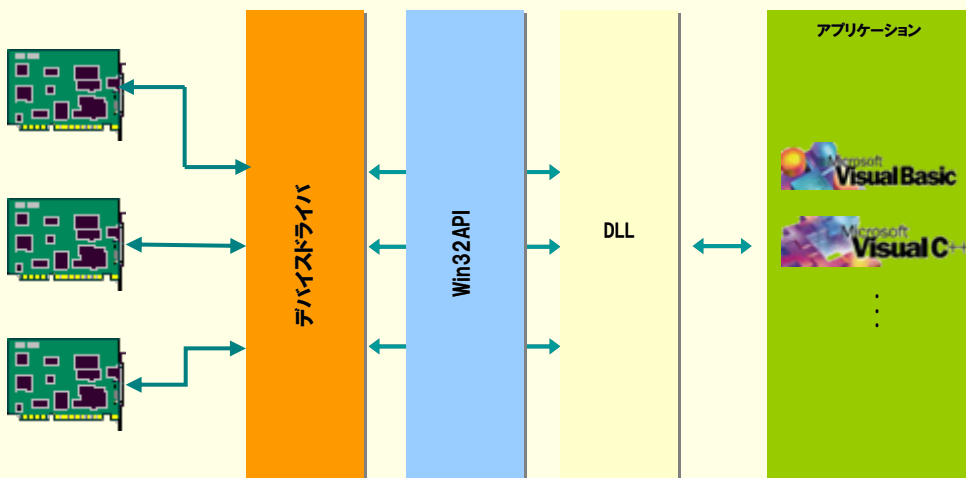
※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しています。実システムにおいては、関数を実行した後にエラー処理のコードを記述します。エラー処理の方法は、各サンプルプログラムを参照してください。

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がデバイスドライバ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。



全チャンネル出力レンジ設定関数『AioSetAoRangeAll』リファレンス

■機能 全チャンネルに対してアナログ出力レンジの設定を行います。

■書式 Visual Basic2005の場合

```
Dim Ret As Integer
Dim Id As Short
Dim AoRange As Short
Ret = AioSetAoRangeAll ( Id , AoRange )
```

■引数 Id : Aiolnit 関数で取得したIDを指定します。

AoRange: アナログ出力レンジを以下の範囲からマクロ (標準モジュールファイル内で設定済みの定数) もしくは数値で指定します。設定できる値はデバイスにより異なります。

レンジ	マクロ	値
±10V	PM10	0
±5V	PM5	1
±2.5V	PM25	2
±1.25V	PM125	3
±1V	PM1	4
±0.625V	PM0625	5
±0.5V	PM05	6
±0.3125V	PM03125	7
±0.25V	PM025	8
±0.125V	PM0125	9
±0.1V	PM01	10
±0.05V	PM005	11
±0.025V	PM0025	12
±0.0125V	PM00125	13

レンジ	マクロ	値
0~10V	P10	50
0~5V	P5	51
0~4.095V	P4095	52
0~2.5V	P25	53
0~1.25V	P125	54
0~1V	P1	55
0~0.5V	P05	56
0~0.25V	P025	57
0~0.1V	P01	58
0~0.05V	P005	59
0~0.025V	P0025	60
0~0.0125V	P00125	61
0~20mA	P20MA	100
4~20mA	P4T020MA	101
1~5V	P1T05	150

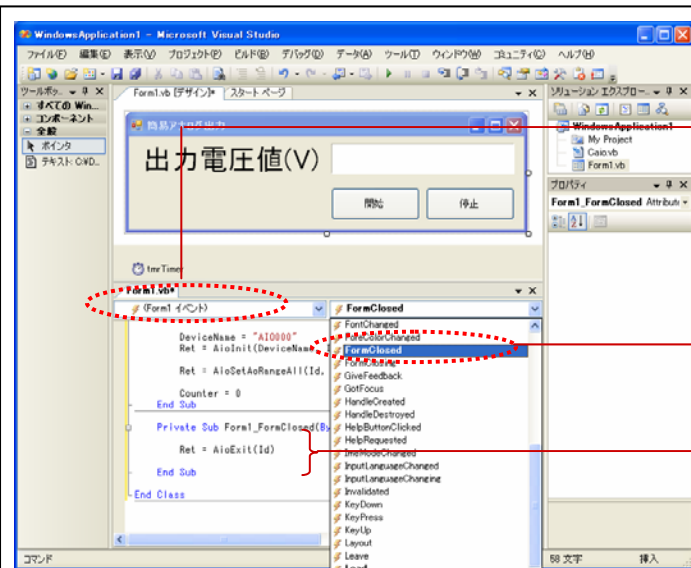
Ret: 終了情報 (戻り値) → 正常終了:0、エラー終了: 0以外 (詳細はヘルプの「戻り値一覧」参照)。

※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しています。
 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 本書にて使用している『ADA16-8/2 (CB) L』は、出力レンジ『±10V固定』です。

4-6-8.プログラム作成手順⑤ (終了処理の追加)

API-AIO (WDM) 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_FormClosedイベント』内で、終了処理関数を実行します。『Form_FormClosedイベント』は『Form』が画面から消去 (クローズド) 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『FormClosed』を選択します。

③ Private Sub Form_FormClosed () 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 Form1_FormClosed ()』から『End Sub』の間に記述してください。

```
Ret = AioExit (Id)
```

‘終了処理 (デバイスハンドル開放)

終了処理関数 『AioExit』 リファレンス

■機能 ドライバの終了処理を行います。この関数は、アプリケーションの終了時に実行します。この関数を実行せずにアプリケーションを終了すると、以降デバイスにアクセスできなくなることがあります。

■書式 Visual Basic2005の場合

```
Dim Id As Short  
Dim Ret As Integer  
Ret = AioExit (Id)
```

■引数 Id: 終了するデバイスハンドルを指定します。AioInit関数で取得した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)。

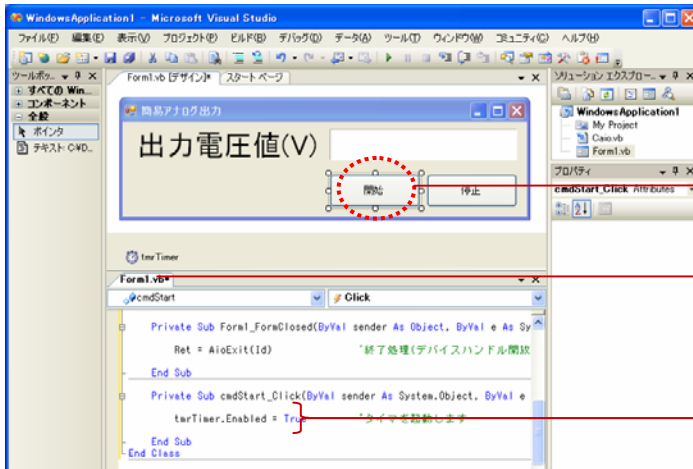
※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しています。実システムにおいては、関数を実行した後にエラー処理のコードを記述します。エラー処理の方法は、各サンプルプログラムを参照してください。

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

4-6-9.プログラム作成手順⑥ (タイマ開始処理の記述)

100msec (0.1秒) 毎にアナログ出力処理を行うために、[4-2.]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[4-2.]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。

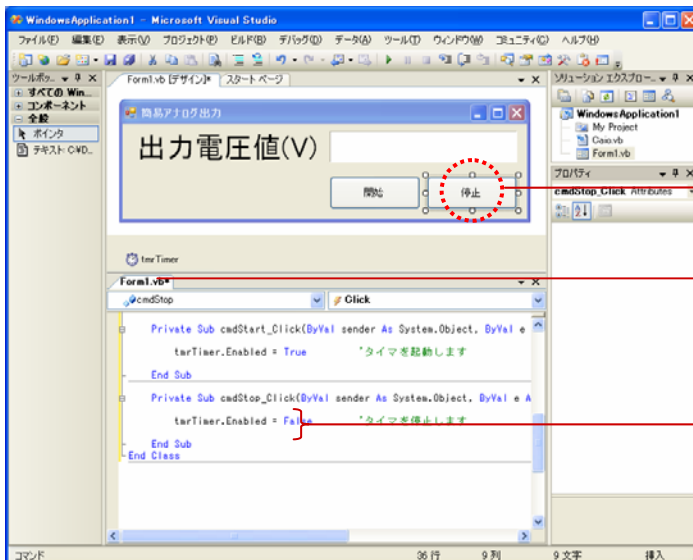
Private Sub cmdStart_Click () からEnd Subまでの間に、下記のコードを記述します。

```
tmrTimer.Enabled = True
```

‘タイマを起動します’

4-6-10.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 () 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’

4-6-11.プログラム作成手順⑧ (アナログ出力処理の追加)

アナログ入出力カードADA16-8/2 (CB) Lの『出力0ch』から、アナログ (電圧) データの出力を行う処理を追加します。アナログ信号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。

100msec (0.1秒) 毎に生成されたサイン波形データが指定したチャンネルから出力されます。

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_Tick () から、End Subの間に記述します。
サイン波形データを最初に生成しアナログ出力関数 (電圧値) を使用して外部に出力します。

Private Sub tmrTimer_Tick () から、End Subの間に下記を記述します。

```

If Counter >= 360 Then          'カウンタが360 (360°) になったら0クリア
    Counter = 0
End If

AoData = 10# * Math.Sin (2 * Math.PI * Counter / 360) 'サイン波 (振幅±10) データ算出

Ret = AioSingleAoEx (Id, 0, AoData)          'データ出力 (アナログ出力)

lblData.Text = Format (AoData, "#0.0000V")    '出力データ (電圧値) を表示

Counter = Counter + 5                    '5° ずつ角度を加算
    
```

アナログ出力 (電圧/電流値) 関数 『AioSingleAoEx』 リファレンス

- 機能 指定チャンネルを1回D/A変換し、変換データを電圧値または電流値で指定します。
- 書式 Visual Basic2005の場合

```

Dim Ret          As Integer
Dim AoData       As Single
Dim AoChannel    As Short
Dim Id           As Short

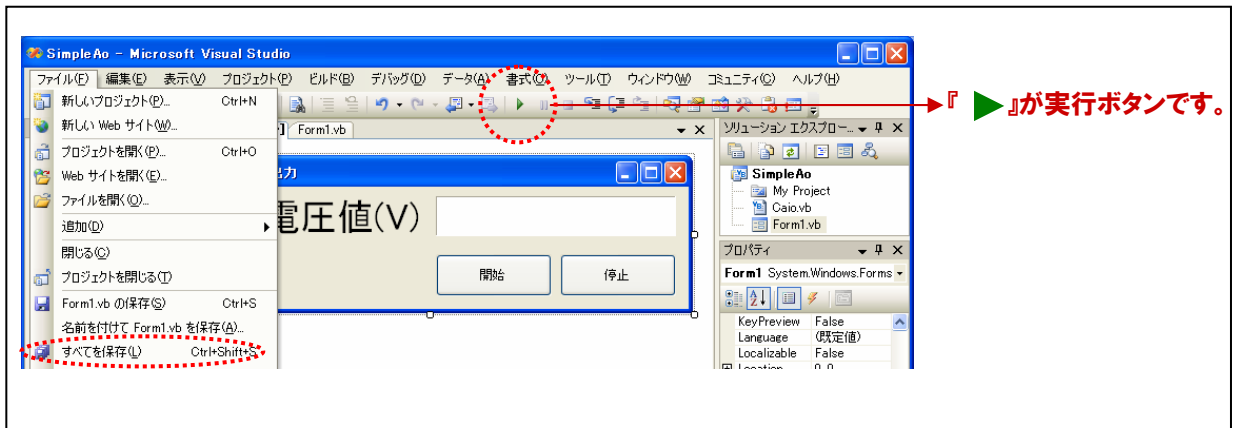
Ret = AioSingleAiEx ( Id , AoChannel , AoData )
    
```

- 引数 Id: AioInit関数で取得したIDを指定します。
- AoChannel: 変換するチャンネルを指定します。
- AoData: 出力するデータが格納された変数を指定します。データは電圧または電流値です。
- 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)。

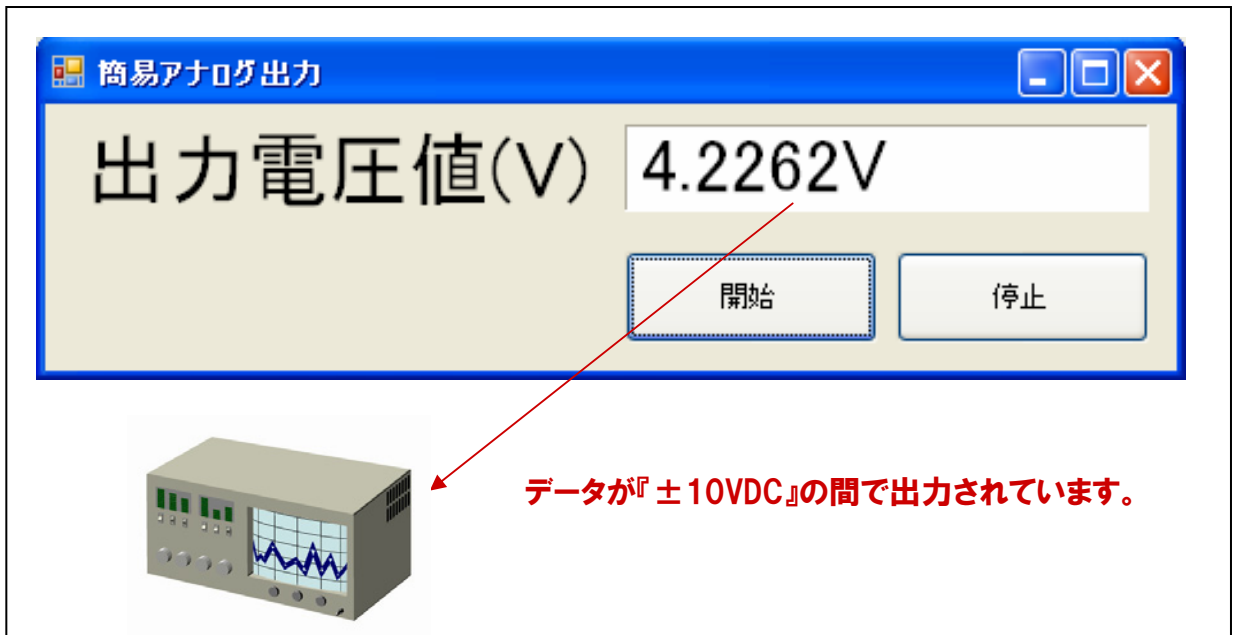
※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しています。実システムにおいては、関数を実行した後にエラー処理のコードを記述します。エラー処理の方法は、各サンプルプログラムを参照してください。

4-6-12.プログラムの実行

- ① 『ファイル』メニューの中から『すべてを保存』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。保存が終わりましたら、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 出力チャンネル0には、デジタルオシロスコープがBNCケーブルによって接続されています。画面上の『開始ボタン』をクリックして見ましょう。±10VDCの振幅で、データが出力されていることがわかります。『停止ボタン』で入力が停止します。



出力されている電圧値をデジタルオシロスコープやデジタルマルチメータで確認して見ると、“-10V”は正常に出力されているのにも関わらず、“+”に関しては、約9.9998...Vまでしか出力されていません。その理由は【第2章 2-5-7.バイナリデータと電圧値の関係（分解能16ビットの場合）】で説明したとおりです。

アナログ入出力カードADA16-8/2 (CB) Lは、分解能16ビットのアナログ出力が行えます。実際に外部へ電圧/電流値を出力するためには、ボード/カードに正しくデータをセットしなければなりません。つまり、プログラム上で例えば+10Vや±0Vと記述していても、実際にボードにセットするデータは、16ビットのバイナリデータでなければなりません。今回使用した簡易アナログ出力関数『AioSingleAoEx』は、出力したいデータを電圧/電流値で指定すると内部でバイナリ変換して、ボードにセットしてくれています。

API-AIO (WDM) には、『AioSingleAo』という関数も提供されています。指定チャンネルを1回D/A変換する機能は同じですが、出力するデータをバイナリ値で指定する関数です。

アナログ出力関数『AioSingleAo』リファレンス

- 機能 指定チャンネルを1回D/A変換します。変換データはバイナリ値で指定します。
- 書式 Visual Basic2005の場合

```
Dim Ret As Integer
Dim AoData As Integer
Dim AoChannel As Short
Dim Id As Short
```

```
Ret = AioSingleAo ( Id , AoChannel , AoData )
```

- 引数 Id: Aiolnit関数で取得したIDを指定します。
AoChannel: 変換するチャンネルを指定します。
AOData: 変換データを格納する変数を指定します。変換データはバイナリです。
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、割愛しています。
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。

API-AIO (WDM) が提供している『AioSingleAoEx』のような高機能関数が提供されていない場合には、出力したい電圧/電流値は、公式を使用して適切なバイナリ(2進数)データに変換してボード/カードへセットします。

バイポーラ(双極)レンジの場合

$$D/A変換データ = \left(\text{電圧 (V)} + \frac{\text{フルスケールレンジ}}{2} \right) \times \frac{\text{分解能}}{\text{フルスケールレンジ}}$$

- ※ フルスケールレンジ: ±10Vレンジの場合は「20」。
- ※ 分解能は、12ビットの場合は「4096」、16ビットの場合は「65536」。

ユニポーラ(単極)レンジの場合

$$D/A変換データ = \text{電圧 (V)} \times \frac{\text{分解能}}{\text{フルスケールレンジ}}$$

- ※ フルスケールレンジ: 0~+10Vレンジの場合は「10」。
- ※ 分解能は、12ビットの場合は「4096」、16ビットの場合は「65536」。

4-6-13. タイマコントロールによる簡易連続アナログ出力プログラムリスト

Dim Ret	As Integer	'戻り値用変数
Dim DeviceName	As String	'デバイス名設定用変数
Dim Id	As Short	'デバイスID格納用変数
Dim AoData	As Single	'出力電圧値格納用変数
Dim Counter	As Short	'カウンタ (サイン波形算出用) 変数

Private Sub Form1_Load (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

DeviceName = "AIO000"	'デバイス名を変数に格納
Ret = AioInit (DeviceName, Id)	'初期化処理
Ret = AioSetAoRangeAll (Id, PM10)	'出力レンジ設定 (±10VDC)
Counter = 0	'サイン波算出用変数0クリア

End Sub

Private Sub Form1_FormClosed (ByVal sender As Object, ByVal e As System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed

Ret = AioExit (Id)	'終了処理 (デバイスハンドル開放)
--------------------	--------------------

End Sub

Private Sub cmdStart_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdStart.Click

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

End Sub

Private Sub cmdStop_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdStop.Click

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

End Sub

Private Sub tmrTimer_Tick (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tmrTimer.Tick

If Counter >= 360 Then Counter = 0 End If	'カウンタが360 (360°) になったら0クリア
AoData = 10# * Math.Sin (2 * Math.PI * Counter / 360)	'サイン波 (振幅±10) データ算出
Ret = AioSingleAoEx (Id, 0, AoData)	'データ出力 (アナログ出力)
lblData.Text = Format (AoData, "#0.0000V")	'出力データ (電圧値) を表示
Counter = Counter + 5	'5° ずつ角度を加算

End Sub

4-7.FIFOメモリを使用した高速連続アナログ出力プログラム

デバイスに搭載されているデバイスバッファメモリ (FIFO形式) にデータをセットして、高速・高精度で外部にアナログ出力を行うプログラムを作成します。

4-7-1.プログラム概要

1000個分のサイン波形データを生成したのち、そのデータをデバイス上のメモリにセットします。変換開始であらかじめ設定した周期 (本プログラムでは1000 μ sec) で高速出力します。アナログ入出力カード『ADA16-8/2 (CB)L』の『アナログ出力0ch』から出力されたサイン波形データをデジタルオシロスコープで確認します。

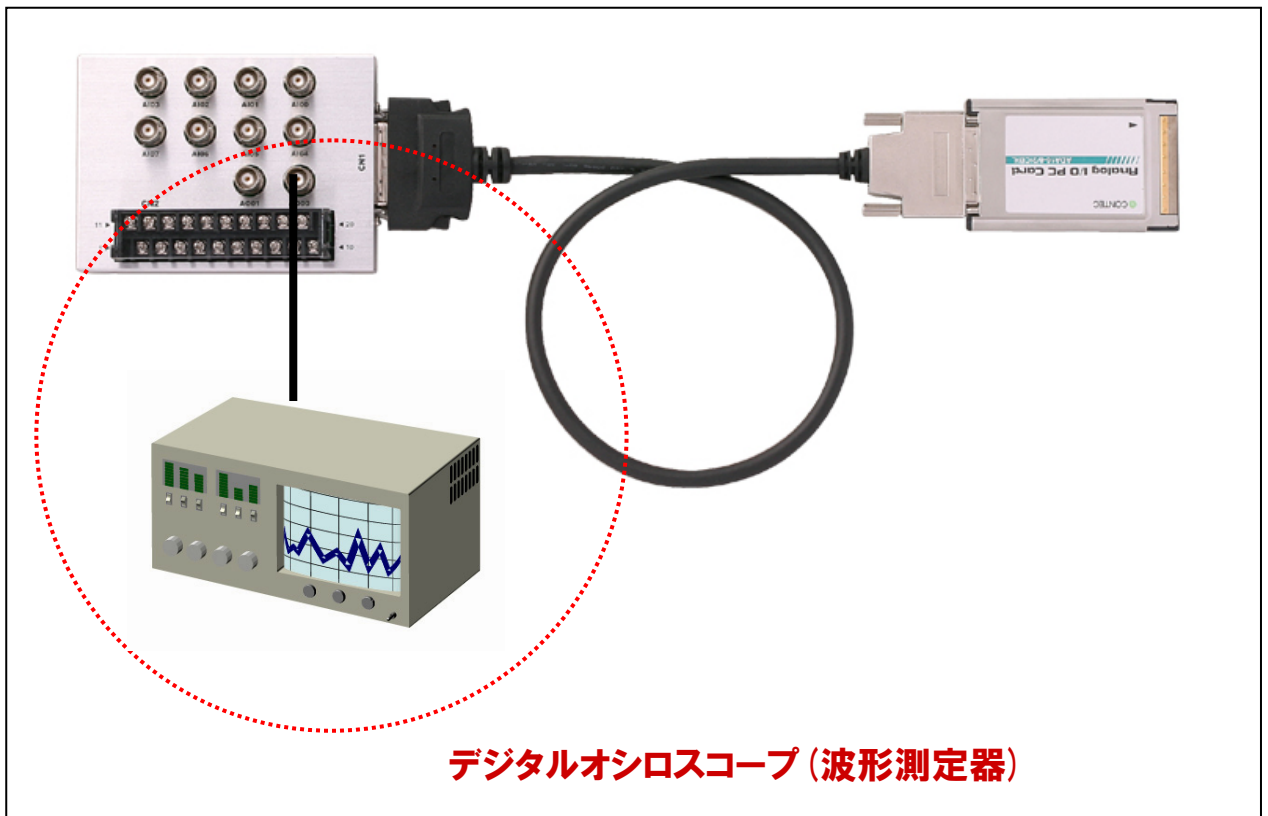


28	1.688
29	1.750
30	1.812
31	1.874
32	1.935
33	1.997
34	2.059
35	2.120
36	2.181
37	2.243
38	2.304
39	2.365
40	2.426
41	2.487
42	2.548
43	2.608
44	2.669
45	2.730

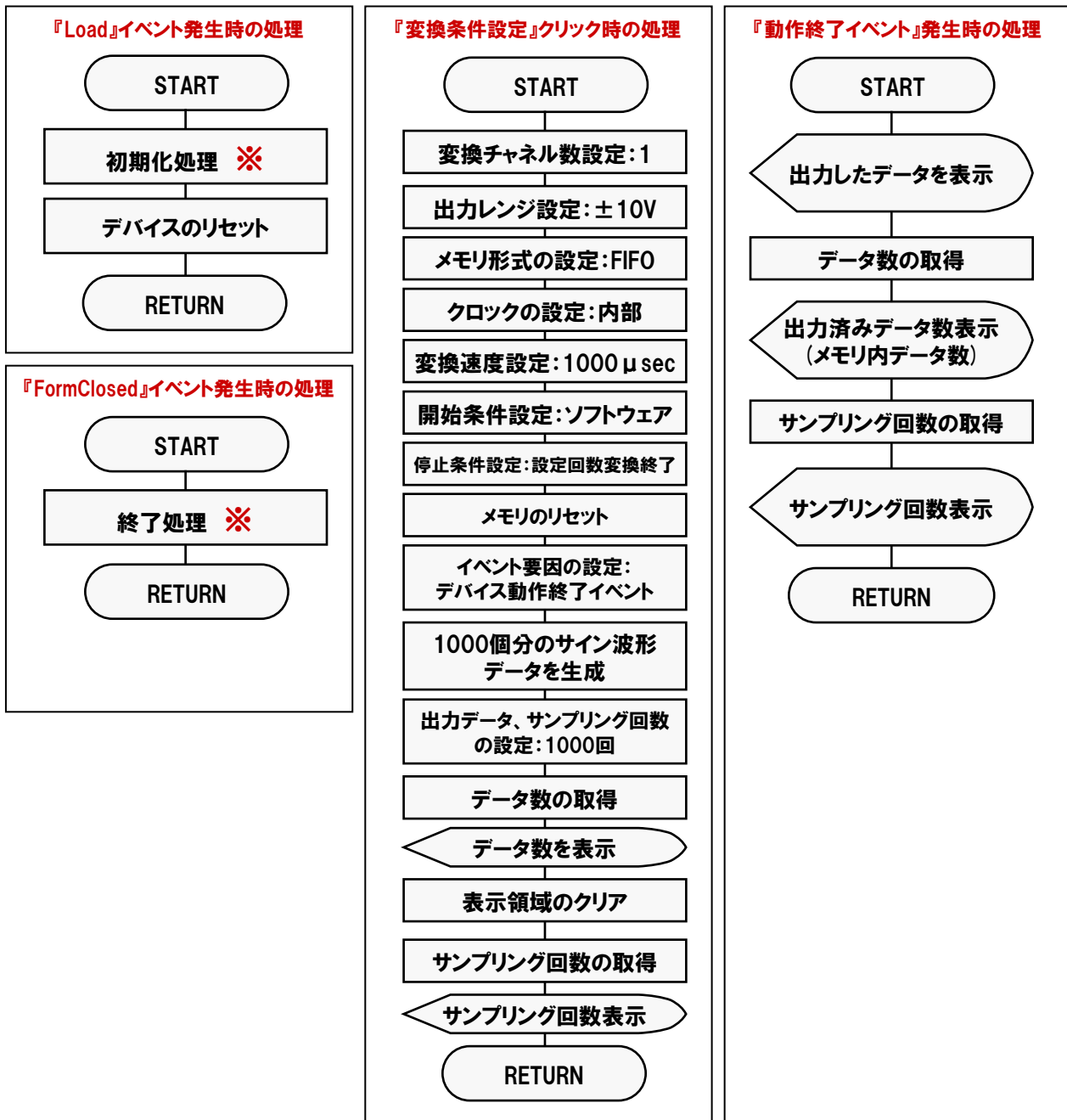
実行画面イメージ

4-7-2.実行環境

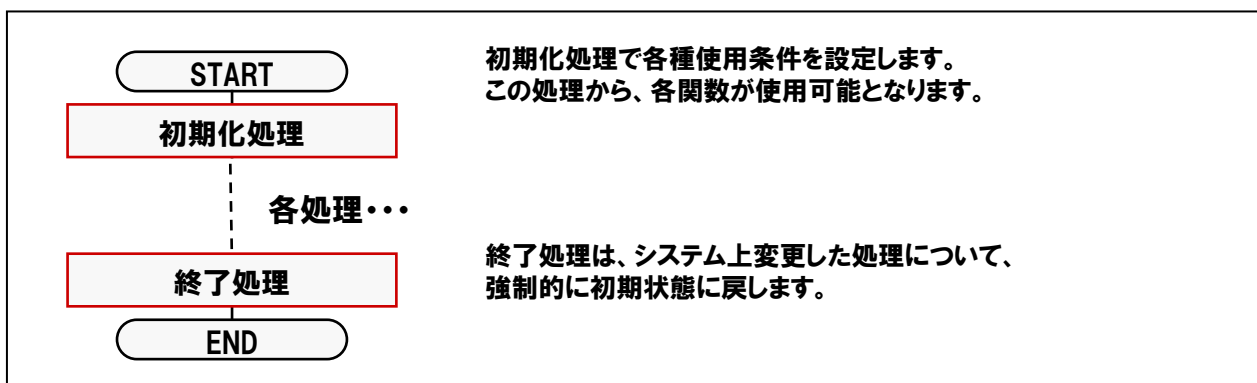
【3-6-3.実習環境の構築】の環境を使用します。使用するデジタルオシロスコープの操作方法に関しては、それぞれの説明書に従ってください。



4-7-3.プログラム フローチャート



※: API-AIO (WDM) の処理体系は、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、その関数を実行します。



4-7-4.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)

[4-2-2.]~[4-2-7.]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
○Text = 『FIFOを使用した高速アナログ出力』
- 変換条件・データ設定ボタン
○Text = 『変換条件・データ設定』
○Name = 『cmdSet』
- データ出力ボタン
○Text = 『データ出力』
○Name = 『cmdStart』
- テキストボックスコントロール
○Name = 『txtData』
○MultiLine = 『True』
○ScrollBars = 『Vertical』
- データ数表示用ラベルコントロール
○Name = 『lblDataNum』
- サンプリング回数表示用ラベルコントロール
○Name = 『lblCount』

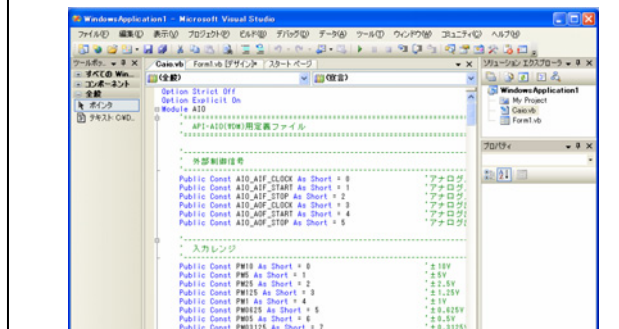
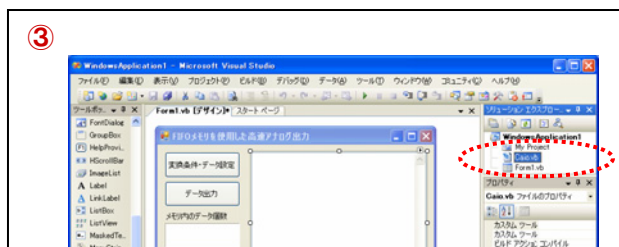
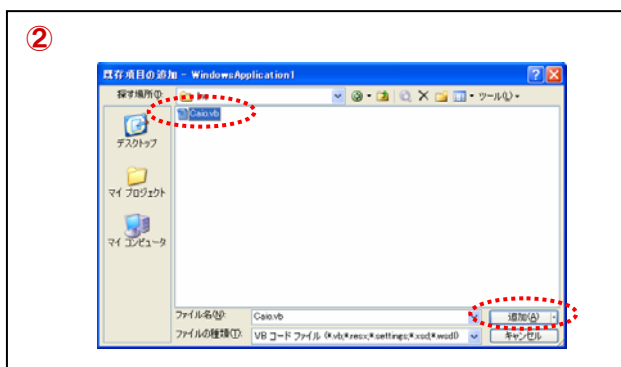
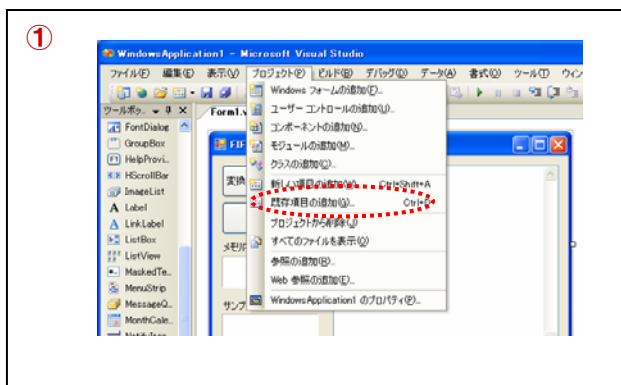
4-7-5.プログラム作成手順② (標準モジュールファイルの追加)

Visual BasicでAPI-AIO (WDM) 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』 - 『既存項目の追加』を選択します。
- ② 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 (Caio.vb) 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) ¥AIOWDM¥Sample¥Inc¥Caio.vb

- ③ 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。



Caio.vb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、定数の宣言などがされているのが分かります。

4-7-6.プログラム作成手順③ (割り込み処理の実現:WndProc)

API-AIO (WDM) で割り込みを実現させるためにWndProc関数を使用します。割り込みのソフト的な処理説明に関しては、【4-4-6】を参照してください。WndProc関数は、Windowsメッセージと内容を取得する関数です。

Visual Basic2005、Visual C#で使用し、API-AIO (WDM) では以下のメッセージを扱います。

アナログ入力メッセージ要因	マクロ	値
AD変換開始条件成立イベント	AIOM_AIE_START	1000H
リピート終了イベント	AIOM_AIE_RPTEND	1001H
デバイス動作終了イベント	AIOM_AIE_END	1002H
指定サンプリング回数格納イベント	AIOM_AIE_DATA_NUM	1003H
指定転送数毎イベント	AIOM_AIE_DATA_TSF	1007H
オーバーフローイベント	AIOM_AIE_OFERR	1004H
サンプリングクロックエラーイベント	AIOM_AIE_SCERR	1005H
A/D変換エラーイベント	AIOM_AIE_ADERR	1006H
アナログ出力メッセージ要因	マクロ	値
DA変換開始条件成立イベント	AIOM_AOE_START	1020H
リピート終了イベント	AIOM_AOE_RPTEND	1021H
デバイス動作終了イベント	AIOM_AOE_END	1022H
指定転送数毎イベント	AIOM_AOE_DATA_TSF	1027H
指定サンプリング回数出力イベント	AIOM_AOE_DATA_NUM	1023H
サンプリングクロックエラーイベント	AIOM_AOE_SCERR	1025H
D/A変換エラーイベント	AIOM_AOE_ADERR	1026H
カウンタメッセージ要因	マクロ	値
動作終了イベント	AIOM_CNTE_END	1040H
動作開始条件成立イベント	AIOM_CNTE_START	1041H
比較カウント一致イベント	AIOM_CNTE_DATA_NUM	1042H
カウントオーバーランイベント	AIOM_CNTE_ORERR	1043H
タイマメッセージ要因	マクロ	値
インターバル成立イベント	AIOM_TME_INT	1060H

Visual Basic2005でのイベントメッセージルーチンは次の書式になります。

`Protected Overrides Sub WndProc (ByRef m As System.Windows.Forms.Message`

◎m.Msg : メッセージ番号が渡されます。

◎m.WParam : 下位2バイトにIDが渡されます。上位2バイトは現在使用しません。

◎m.LParam : イベントごとに固有のパラメータが渡されます。本書では『デバイス動作終了イベント』を使用しますので、下表のとおり『m.LParam』引数には『現在のサンプリング回数』が渡されてきます。

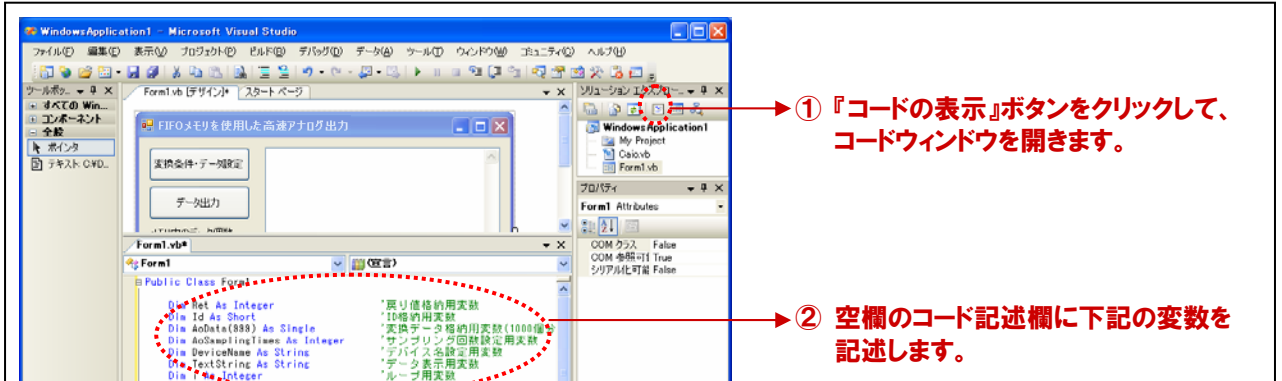
イベント要因	パラメータ
AD変換開始条件成立イベント	なし
リピート終了イベント	現在のリピート回数
デバイス動作終了イベント	現在のサンプリング回数
指定サンプリング回数出力イベント	現在のサンプリング回数
指定転送数毎イベント	現在の転送回数
オーバーフローエラーイベント	現在のサンプリング回数
サンプリングクロックエラーイベント	現在のサンプリング回数
AD変換エラーイベント	現在のサンプリング回数

4-7-7.プログラム作成手順④ (変数の追加)

簡易オシロスコープ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』 - 『プログラム』 - 『CONTEC API-PAC (W32)』 - 『AIOWDM』 - 『API-AIO (WDM) 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。

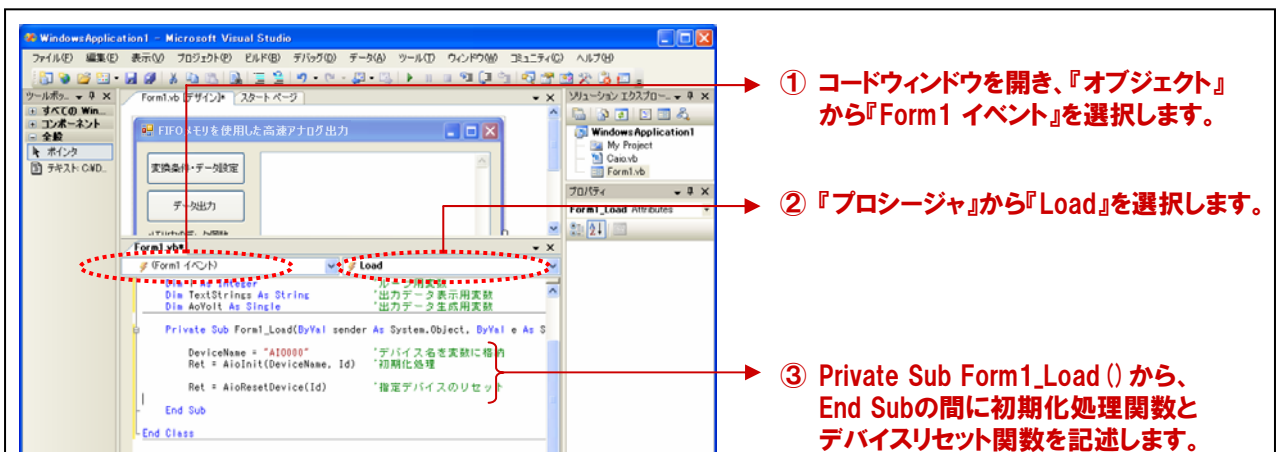
② 空欄のコード記述欄に下記の変数を記述します。

Dim Ret	As Integer	'戻り値格納用変数
Dim Id	As Short	'ID格納用変数
Dim AoData (999)	As Single	'変換データ格納用変数 (1000個分の配列変数)
Dim AoSamplingTimes	As Integer	'サンプリング回数設定用変数
Dim AoSamplingCount	As Integer	'サンプリング回数取得用変数
Dim DeviceName	As String	'デバイス名設定用変数
Dim TextString	As String	'データ表示用変数
Dim i	As Short	'ループ用変数
Dim TextStrings	As String	'出力データ表示用変数
Dim AoVolt	As Single	'出力データ生成用変数

4-7-8.プログラム作成手順⑤ (初期化処理とアナログ出力デバイスリセットの追加)

API-AIO (WDM) 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、変数を初期化の際に使われます。

また、アナログ入力デバイスのリセットを行います。



① コードウィンドウを開き、『オブジェクト』から『Form1 イベント』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 Form1_Load () から、End Subの間に初期化処理関数とデバイスリセット関数を記述します。

下記のコードを『Private Sub Form1_Load ()』から『End Sub』の間に記述してください。

DeviceName = "AI0000"	'デバイス名を変数に格納
Ret = AioInit (DeviceName, Id)	'初期化処理
Ret = AioResetDevice (Id)	'指定デバイスのリセット

※:各関数で設定しているパラメータに関する詳細は、次項からの『関数リファレンス』で解説します。

初期化処理関数『AioInIt』リファレンス

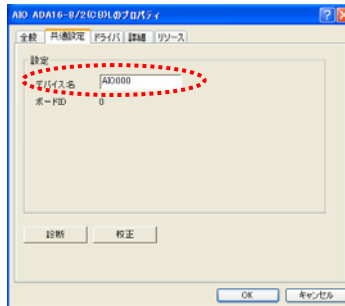
- 機能 デバイスファイルを作成し、以降デバイスを使用可能にします。デバイスにアクセスするには、まずこの関数を実行する必要があります。この関数がプロセスで初めて実行された時には、内部のパラメータがすべてデフォルト値に設定されます。デバイス内にレジスタを持つ場合はそのレジスタもデフォルト値に設定されます。AioExit関数を実行せずに続けてAioInItを実行しても、内部のパラメータはデフォルトに戻りません。内部パラメータをデフォルト値に戻すには、AioResetDeviceを使用します。

- 書式 Visual Basic2005の場合

```
Dim Ret As Integer
Dim DeviceName As String
Dim Id As Short
Ret = AioInIt ( DeviceName , Id )
```

- 引数 DeviceName: デバイスマネージャのプロパティページ、または設定ツールで設定したデバイス名を指定します。
Id: ID (デバイスハンドル)を受け取る変数を指定します。以降の関数は、この変数に格納された値を用いてアクセスできます。

デバイス名に関して



3-5.手順 ④:ドライバソフトウェアの初期設定の項目で設定した、デバイス名を使用します。

本書では、デフォルト値『AI0000』を使用します。

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、割愛しています。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。

デバイスリセット処理関数『AioResetDevice』リファレンス

- 機能 デバイスのリセット、ドライバの初期化を行います。

- 書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Ret = AioResetDevice ( Id )
```

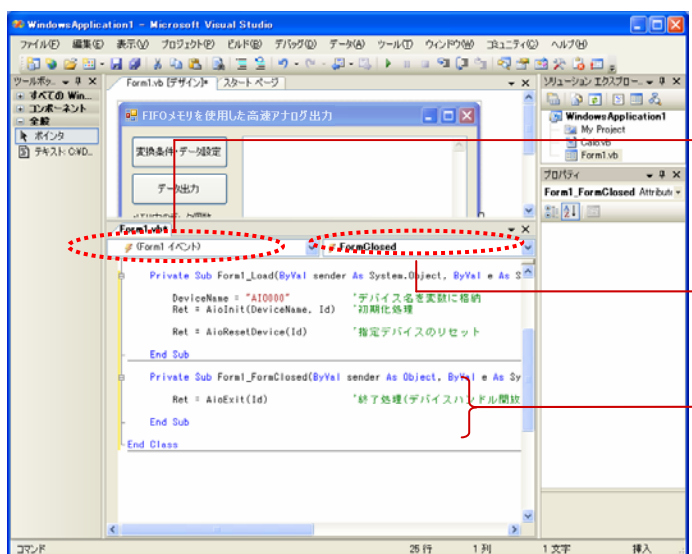
- 引数 Id: AioInIt関数で取得したIDを指定します。

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、割愛しています。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。

4-7-9.プログラム作成手順⑥ (終了処理の追加)

API-AIO (WDM) 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_FormClosedイベント』内で、終了処理関数を実行します。『Form_FormClosedイベント』は『Form』が画面から消去 (クローズド) 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『Form1 イベント』を選択します。

② 『プロシージャ』から『FormClosed』を選択します。

③ Private Sub Form1_FormClosed () 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 Form1_FormClosed ()』から『End Sub』の間に記述してください。

```
Ret = AioExit (Id)
```

‘終了処理 (デバイスハンドル開放)

終了処理関数 『AioExit』 リファレンス

■機能 ドライバの終了処理を行います。この関数は、アプリケーションの終了時に実行します。この関数を実行せずにアプリケーションを終了すると、以降デバイスにアクセスできなくなることがあります。

■書式 Visual Basic2005の場合

```
Dim Id As Short  
Dim Ret As Integer  
Ret = AioExit (Id)
```

■引数 Id: 終了するデバイスハンドルを指定します。AioInit関数で取得したIDを指定します。

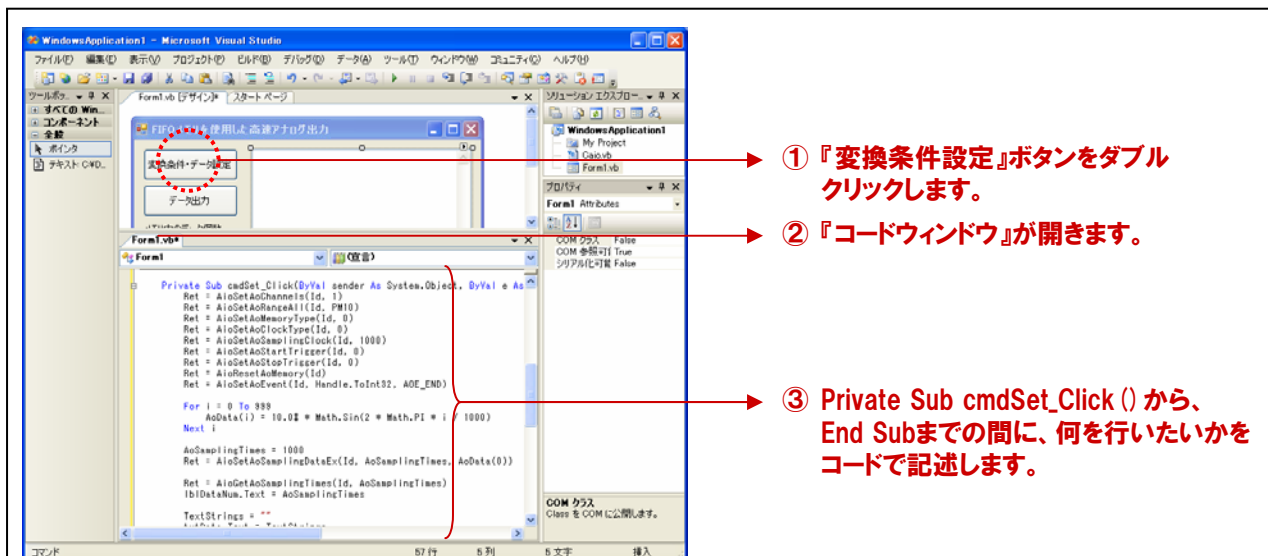
Ret: 終了情報 (戻り値) → 正常終了:0、エラー終了: 0以外 (詳細はヘルプの「戻り値一覧」参照)。

※ 本書では、戻り値の確認 (エラー処理) は、誌面の都合上、割愛しています。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、ドライバの各関数は実行できません。

4-7-10.プログラム作成手順⑦(変換条件設定処理の追加)

メモリ形式 (FIFOまたはRING) や変換クロック、変換速度などアナログ入出力デバイスにセットするアナログ入出力変換条件を設定します。各種設定は、関数を実行するだけで完了です。



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

```

Ret = AioSetAoChannels (Id, 1) '出力チャンネル数設定 (1チャンネルのみ)
Ret = AioSetAoRangeAll (Id, PM10) '出力レンジ設定 (PM10:±10V)
Ret = AioSetAoMemoryType (Id, 0) 'メモリ形式の設定: (0:FIFO)
Ret = AioSetAoClockType (Id, 0) 'クロック種類の設定: (0:内部)
Ret = AioSetAoSamplingClock (Id, 1000) '変換速度の設定:1000 μsec
Ret = AioSetAoStartTrigger (Id, 0) '開始条件の設定:ソフトウェア
Ret = AioSetAoStopTrigger (Id, 0) '停止条件の設定:設定回数変換終了
Ret = AioResetAoMemory (Id) 'メモリのリセット
Ret = AioSetAoEvent (Id, Handle.ToInt32, AOE_END) 'イベント要因の設定:デバイス動作終了イベント

For j = 0 To 999
    AoData (i) = 10# * Math.Sin (2 * Math.PI * i / 1000) 'サイン波形データ生成 (1000個分)
Next i

AoSamplingTimes = 1000
Ret = AioSetAoSamplingDataEx (Id, AoSamplingTimes, AoData (0)) '出力データ、サンプリング回数の設定:1000回

Ret = AioGetAoSamplingTimes (Id, AoSamplingTimes) 'データ数を取得
lblDataNum.Text = AoSamplingTimes 'データ数を表示

TextStrings = "" '表示領域のクリア
txtData.Text = TextStrings

Ret = AioGetAoSamplingCount (Id, AoSamplingCount) 'サンプリング回数を取得
lblCount.Text = AoSamplingCount 'サンプリング回数を表示
    
```

※:各関数で設定しているパラメータに関する詳細は、次項からの『関数リファレンス』で解説します。

出力チャンネル数設定処理関数『AioSetAoChannels』リファレンス

■機能 変換に使用するアナログ出力チャンネル数の設定を行います。

■書式 Visual Basic2005の場合

```
Dim Id          As Short
Dim Ret         As Integer
Dim AoChannels As Short
Ret = AioSetAoChannels ( Id , AoChannels )
```

■引数 Id: Aiolnit関数で取得したIDを指定します。

AoChannels: 変換に使用するチャンネル数を指定します。

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、割愛しています。
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。

全チャンネル出力レンジ設定関数『AioSetAoRangeAll』リファレンス

■機能 全チャンネルに対してアナログ出力レンジの設定を行います。

■書式 Visual Basic2005の場合

```
Dim Ret         As Integer
Dim Id          As Short
Dim AoRange     As Short
Ret = AioSetAoRangeAll ( Id , AoRange )
```

■引数 Id : Aiolnit 関数で取得したIDを指定します。

AoRange: アナログ出力レンジを以下の範囲からマクロ(標準モジュールファイル内で設定済みの定数)もしくは数値で指定します。設定できる値はデバイスにより異なります。

レンジ	マクロ	値
±10V	PM10	0
±5V	PM5	1
±2.5V	PM25	2
±1.25V	PM125	3
±1V	PM1	4
±0.625V	PM0625	5
±0.5V	PM05	6
±0.3125V	PM03125	7
±0.25V	PM025	8
±0.125V	PM0125	9
±0.1V	PM01	10
±0.05V	PM005	11
±0.025V	PM0025	12
±0.0125V	PM00125	13

レンジ	マクロ	値
0~10V	P10	50
0~5V	P5	51
0~4.095V	P4095	52
0~2.5V	P25	53
0~1.25V	P125	54
0~1V	P1	55
0~0.5V	P05	56
0~0.25V	P025	57
0~0.1V	P01	58
0~0.05V	P005	59
0~0.025V	P0025	60
0~0.0125V	P00125	61
0~20mA	P20MA	100
4~20mA	P4T020MA	101
1~5V	P1T05	150

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、割愛しています。
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 本書にて使用している『ADA16-8/2(CB)L』は、出力レンジは『±10V固定』です。

メモリ形式設定処理関数『AioSetAoMemoryType』リファレンス

■機能 データ格納用メモリ形式の設定を行います。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AoMemoryType As Short
Ret = AioSetAoMemoryType ( Id , AoMemoryType )
```

■引数 Id: Aiolnit関数で取得したIDを指定します。

AoMemoryType: 『FIFO = 0』 『RING = 1』。

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、割愛しています。
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 本書では転送方式にデバイスバッファを使用しています。ユーザーバッファでは引数の意味合いが異なります。

クロック種類設定処理関数『AioSetAoClockType』リファレンス

■機能 クロックの種類を取得します。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AoClockType As Short
Ret = AioSetAoClockType ( Id , AoClockType )
```

■引数 Id: Aiolnit関数で取得したIDを指定します。

AoClockType: 『内部クロック = 0』 『外部クロック = 1』 『イベントコントローラ = 10』。

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、割愛しています。
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 本書で使用しているADA16-8/2 (CB) Lは、イベントコントローラは使用できません。詳細はヘルプを参照してください。

変換速度設定処理関数『AioSetAoSamplingClock』リファレンス

■機能 内部クロックを使用する場合に、変換速度の設定を行います。内部クロックを使用しない場合には実行する必要はありません。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AoSamplingClock As Single
Ret = AioSetAoSamplingClock ( Id , AoSamplingClock )
```

■引数 Id: Aiolnit関数で取得したIDを指定します。

AoSamplingClock: 変換速度を μ sec単位で指定します。デバイスにより設定できる範囲は異なります。

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、割愛しています。
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 本書で使用しているADA16-8/2 (CB) Lは、10~107374182の設定が可能です (初期値:1000)。
詳細はヘルプを参照してください。

変換開始条件設定処理関数『AioSetAoStartTrigger』リファレンス

■機能 変換開始条件の設定を行います。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AoStartTrigger As Short
Ret = AioSetAoStartTrigger ( Id , AoStartTrigger )
```

■引数 Id: AiOnit関数で取得したIDを指定します。

AoStartTrigger: 変換開始条件を以下の範囲から設定します。デバイスにより設定できる値は異なります。

0	ソフトウェア
1	外部トリガ立ち上がり
2	外部トリガ立ち下がり
10	イベントコントローラ出力

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、割愛しています。
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 本書で使用しているADA16-8/2 (CB) Lは、AiStartTriggerは0、1、2の設定が可能です。

変換停止条件設定処理関数『AioSetAoStopTrigger』リファレンス

■機能 クロックの種類を取得します。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AoStopTrigger As Short
Ret = AioSetAoStopTrigger ( Id , AoStopTrigger )
```

■引数 Id: AiOnit関数で取得したIDを指定します。

AoStopTrigger: 変換停止条件を以下の範囲から設定します。デバイスにより設定できる値は異なります。

0	設定回数変換終了
1	外部トリガ立ち上がり
2	外部トリガ立ち下がり
10	イベントコントローラ出力

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、割愛しています。
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 本書で使用しているADA16-8/2 (CB) Lは、AiStopTriggerは0、1、2の設定が可能です。

メモリリセット処理関数 『AioResetAoMemory』 リファレンス

■機能 デバイスメモリ、またはドライバメモリをリセットします。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Ret = AioResetAoMemory ( Id )
```

■引数 Id: Aiolnit関数で取得した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、割愛しています。
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。

イベント処理関数 『AioSetAoEvent』 リファレンス

■機能 アナログ出力に関するWindowメッセージ通知のイベント要因を設定します。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim hWnd As Integer
Dim AoEvent As Integer
Ret = AioSetAoEvent ( Id , hWnd , AoEvent )
```

■引数 Id: Aiolnit関数で取得したIDを指定します。

hWnd: Windowハンドルを指定します。Visual BasicとC#の場合、CMessageのハンドルを指定します。

AoEvent: イベント要因を以下の範囲からマクロもしくは数値で指定します。AoEventはビット単位で以下のような意味を持ち、これらを組み合わせて指定可能です。デバイスバッファ使用時とユーザーバッファ使用時で使用可能なイベント要因が異なります。

イベント要因	デバイスバッファ使用時	ユーザーバッファ使用時	マクロ	値
DA変換開始条件成立イベント	○	○	AOE_START	0000002H
リビート終了イベント	○	○	AOE_RPTEND	0000010H
デバイス動作終了イベント	○	○	AOE_END	0000020H
指定サンプリング回数出力イベント	○	×	AOE_DATA_NUM	0000080H
指定転送数毎イベント	×	○	AOE_DATA_TSF	0000100H
サンプリングクロックエラーイベント	○	○	AOE_SCERR	0002000H
DA変換エラーイベント	○	○	AOE_DAERR	0004000H

この関数で設定されたイベント要因は、イベントメッセージルーチンにメッセージとして通知されます。メッセージの種類は、【4-7-6.】を参照してください。

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、割愛しています。
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 本書で使用しているADA16-8/2 (CB) Lでは、デバイスバッファ形式を使用しています。

データ設定処理関数『AioSetAoSamplingDataEx』リファレンス

■機能 出力データを設定します。出力データは電圧または電流で指定します。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AoSamplingTimes As Integer
Dim AoData() As Single
Ret = AioSetAoSamplingDataEx ( Id , AoSamplingTimes , AoData ( 0 ) )
```

■引数 Id: Aiolnit関数で取得したIDを指定します。

AoSamplingTimes: 出力データのサンプリング回数を設定します。

AoData(): 出力データを格納した配列を指定します。データは電圧または電流で指定します。デバイスにより設定できる値は異なります。

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、割愛しています。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。

サンプリング回数取得関数『AioGetAoSamplingTimes』リファレンス

■機能 サンプリング数を取得します。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AoSamplingTimes As Integer
Ret = AioGetAoSamplingTimes ( Id , AoSamplingTimes )
```

■引数 Id: Aiolnit関数で取得したIDを指定します。

AoSamplingTimes: サンプリング数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、割愛しています。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。

サンプリング回数取得関数『AioGetAoSamplingCount』リファレンス

■機能 アナログ出力のサンプリング回数を取得します。

■書式 Visual Basic2005の場合

```
Dim Id As Short
Dim Ret As Integer
Dim AoSamplingCount As Integer
Ret = AioGetAoSamplingCount ( Id , AoSamplingCount )
```

■引数 Id: Aiolnit関数で取得したIDを指定します。

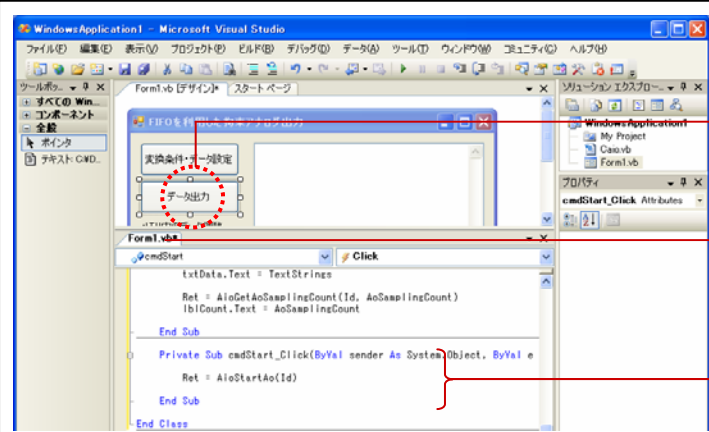
AoSamplingCount: サンプリング回数を格納する配列を指定します。

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、割愛しています。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。

4-7-11.プログラム作成手順⑧ (アナログ出力開始処理の追加)

設定された条件に基づいてD/A変換を開始し、データ出力を行います。



① 『データ出力』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 () 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。

```
Ret = AioStartAo (Id) 'D/A変換を開始します
```

D/A変換開始処理関数 『AioStartAo』 リファレンス

■機能 設定された条件に基づいてD/A変換を開始します。

■書式 Visual Basic2005の場合

```
Dim Id As Short  
Dim Ret As Integer  
Ret = AioStartAo ( Id )
```

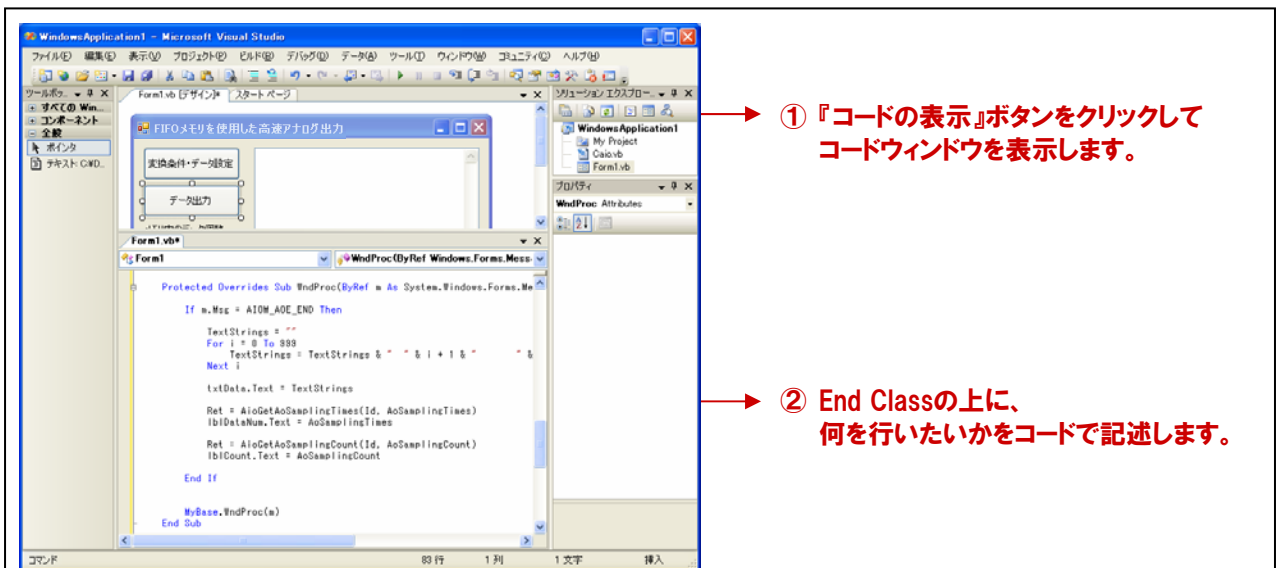
■引数 Id: Aiolnit関数で取得したIDを指定します。

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、割愛しています。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。

4-7-12.プログラム作成手順⑨ (イベント発生時処理の追加)

設定した条件で変換動作が終了すると、デバイスは割り込み信号を発信し、デバイスドライバがそれを受け取り、アプリケーションへ決められたメッセージを発信します。そのメッセージをCmessageコントロールが受け取り、イベントを発生させます。そのイベント発生時処理の中でデータ取得と表示の処理を行います。



End Classの上に、下記のコードを記述します。

```
Protected Overrides Sub WndProc (ByRef m As System.Windows.Forms.Message)
```

```
    If m.Msg = AIOM_AOE_END Then
```

```
        TextStrings = ""                '出力データを表示
```

```
        For i = 0 To 999
```

```
            TextStrings = TextStrings & " " & i + 1 & "    " & Format (AoData (i), "0.000") & vbCrLf
```

```
        Next i
```

```
        txtData.Text = TextStrings
```

```
        Ret = AioGetAoSamplingTimes (Id, AoSamplingTimes)
```

```
        lblDataNum.Text = AoSamplingTimes
```

'データ数を取得

'データ数を表示

```
        Ret = AioGetAoSamplingCount (Id, AoSamplingCount)
```

```
        lblCount.Text = AoSamplingCount
```

'サンプリング回数を取得

'サンプリング回数を表示

```
    End If
```

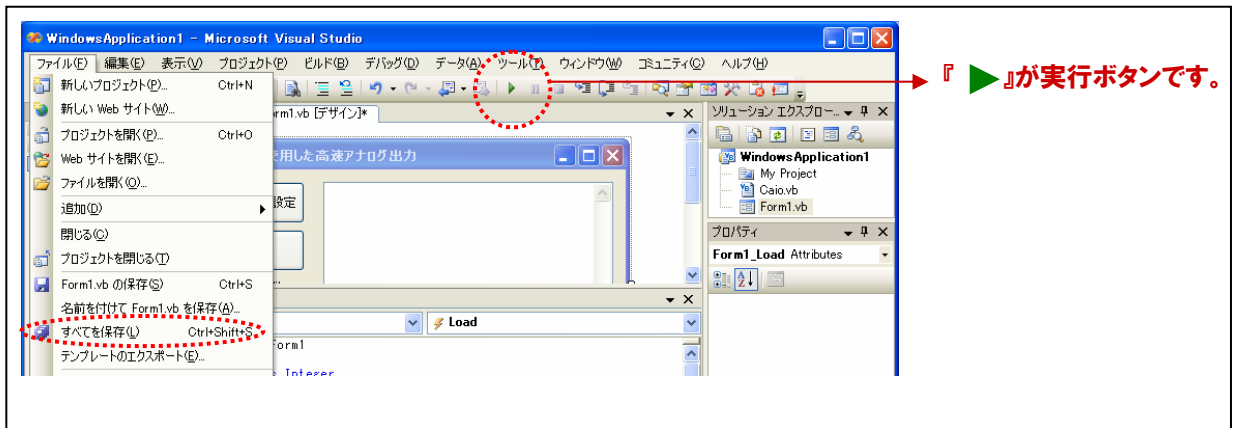
```
    MyBase.WndProc (m)
```

```
End Sub
```

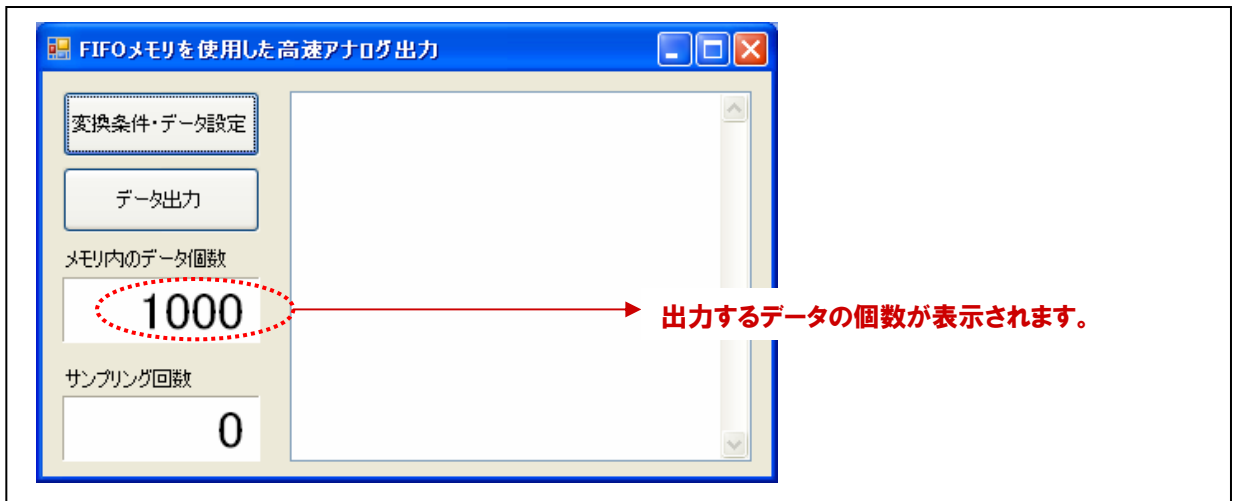
解説： イベントの引数『Message』には、変換が正常に終了した場合、先の変換条件設定で設定したメッセージが渡されてきます。その判断にIf～End If構文を使用しています。For～Next構文を使用して、出力したデータをテキストボックスに表示します。また、メモリ内のデータ個数とサンプリングを行った回数を取得し、表示しています。If～End If構文、For～Next構文に関しては、Visual Basicの説明書を参照願います。

4-7-13.プログラムの実行

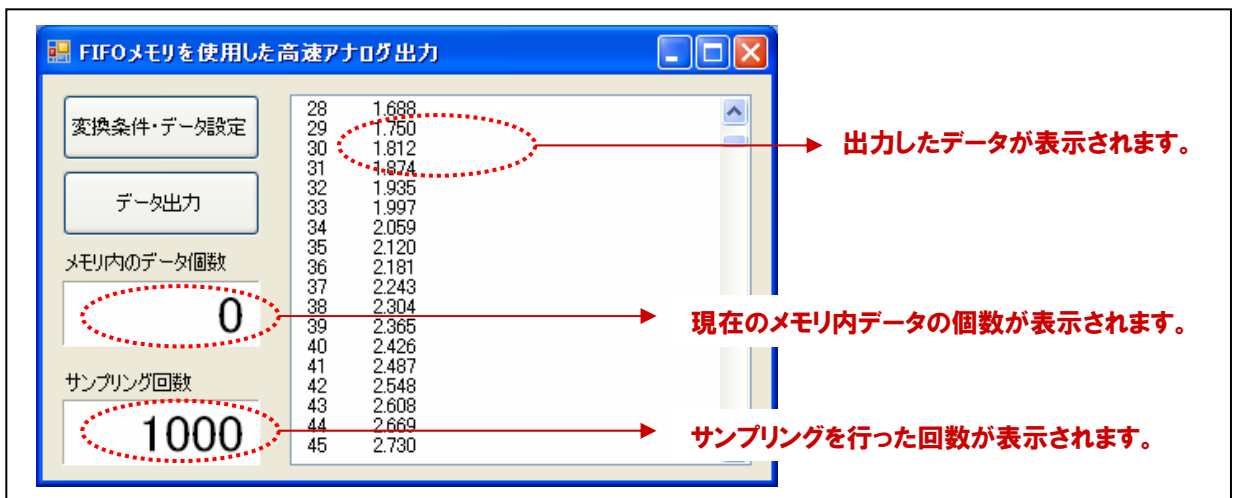
- ① 『ファイル』メニューの中から『名前を付けてプロジェクトの保存』を選択して、任意の場所に、任意のプロジェクト名称にて今回作成したプロジェクトを保存します。保存が終わりましたら、ツールバーの実行ボタンをクリックし、画面上の『開始ボタン』をクリックして実行してみましょう。



- ② 出力チャンネル0には、デジタルオシロスコープがBNCケーブルによって接続されています。画面上の『変換条件・データ設定ボタン』をクリックして見ましょう。この時点で、デバイスに変換条件およびバッファメモリ内に1000個分のデータがセットされます。



- ③ 次に『データ出力ボタン』をクリックします。このタイミングでD/A変換が開始され、変換動作が終了(本プログラムでは1000回のサンプリングが終了した時点)すると、イベントが発生し、実際に出力したデータがテキストボックスに表示されます。また、現在のメモリ内のデータ個数および、サンプリングを行った回数が表示されます。



4-7-14. FIFOメモリを使用した高速連続アナログ出力プログラムリスト

Dim Ret	As Integer	'戻り値格納用変数
Dim Id	As Short	'ID格納用変数
Dim AoData (999)	As Single	'変換データ格納用変数 (1000個分の配列変数)
Dim AoSamplingTimes	As Integer	'サンプリング回数設定用変数
Dim AoSamplingCount	As Integer	'サンプリング回数取得用変数
Dim DeviceName	As String	'デバイス名設定用変数
Dim TextString	As String	'データ表示用変数
Dim i	As Short	'ループ用変数
Dim TextStrings	As String	'出力データ表示用変数
Dim AoVolt	As Single	'出力データ生成用変数

Private Sub Form1_Load (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

DeviceName = "AIO000"	'デバイス名を変数に格納
Ret = AioInit (DeviceName, Id)	'初期化处理
Ret = AioResetDevice (Id)	'指定デバイスのリセット

End Sub

Private Sub Form1_FormClosed (ByVal sender As Object, ByVal e As System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed

Ret = AioExit (Id)	'終了処理 (デバイスハンドル開放)
--------------------	--------------------

End Sub

Private Sub cmdSet_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdSet.Click

Ret = AioSetAoChannels (Id, 1)	'出力チャンネル数設定 (1チャンネルのみ)
Ret = AioSetAoRangeAll (Id, PM10)	'出力レンジ設定 (PM10: ±10V)
Ret = AioSetAoMemoryType (Id, 0)	'メモリ形式の設定: (0:FIFO)
Ret = AioSetAoClockType (Id, 0)	'クロック種類の設定: (0:内部)
Ret = AioSetAoSamplingClock (Id, 1000)	'変換速度の設定: 1000 μsec
Ret = AioSetAoStartTrigger (Id, 0)	'開始条件の設定: ソフトウェア
Ret = AioSetAoStopTrigger (Id, 0)	'停止条件の設定: 設定回数変換終了
Ret = AioResetAoMemory (Id)	'メモリのリセット
Ret = AioSetAoEvent (Id, Handle.ToInt32, AOE_END)	'イベント要因の設定: デバイス動作終了イベント
For i = 0 To 999	
AoData (i) = 10# * Math.Sin (2 * Math.PI * i / 1000)	'サイン波形データ生成 (1000個分)
Next i	
AoSamplingTimes = 1000	
Ret = AioSetAoSamplingDataEx (Id, AoSamplingTimes, AoData (0))	'出力データ、サンプリング回数の設定: 1000回
Ret = AioGetAoSamplingTimes (Id, AoSamplingTimes)	'データ数を取得
lblDataNum.Text = AoSamplingTimes	'データ数を表示
TextStrings = ""	'表示領域のクリア
txtData.Text = TextStrings	
Ret = AioGetAoSamplingCount (Id, AoSamplingCount)	'サンプリング回数を取得
lblCount.Text = AoSamplingCount	'サンプリング回数を表示

End Sub

```
Private Sub cmdStart_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdStart.Click
```

```
    Ret = AioStartAo (Id)                                     'D/A変換を開始します
```

```
End Sub
```

```
Protected Overrides Sub WndProc (ByRef m As System.Windows.Forms.Message)
```

```
    If m.Msg = AIOM_AOE_END Then
```

```
        TextStrings = ""                                     '出力データを表示
        For i = 0 To 999
            TextStrings = TextStrings & " " & i + 1 & "    " & Format (AoData (i), "0.000") & vbCrLf
        Next i
```

```
        txtData.Text = TextStrings
```

```
        Ret = AioGetAoSamplingTimes (Id, AoSamplingTimes) 'データ数を取得
        lblDataNum.Text = AoSamplingTimes                 'データ数を表示
```

```
        Ret = AioGetAoSamplingCount (Id, AoSamplingCount) 'サンプリング回数を取得
        lblCount.Text = AoSamplingCount                  'サンプリング回数を表示
```

```
    End If
```

```
    MyBase.WndProc (m)                                     'ベースクラスのWndProc処理
```

```
End Sub
```


4-8.ドライバソフトウェア提供関数一覧

高性能アナログ入出力ドライバ『API-AIO (WDM)』は、弊社製アナログ入出力ボード/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共通関数一覧

共通	
AioInit	初期化処理
AioExit	終了処理
AioResetProcess	プロセスのリセット
AioResetDevice	デバイスとドライバの初期化
AioGetErrorString	エラー内容の取得
AioQueryDeviceName	使用可能なデバイスの一覧を取得
AioGetDeviceType	デバイスの種類を取得
AioSetControlFilter	外部制御信号に対してデジタルフィルタを設定
AioGetControlFilter	外部制御信号に設定されているデジタルフィルタを取得

②アナログ入力関数一覧

簡易入力	
AioSingleAi	指定チャンネルを1回バイナリ値で入力
AioSingleAiEx	指定チャンネルを1回電圧または電流値で入力
AioMultiAi	複数チャンネルを1回バイナリ値で入力
AioMultiAiEx	複数チャンネルを1回電圧または電流値で入力
分解能	
AioGetAiResolution	分解能を取得
入力方式	
AioSetAiInputMethod	入力方式の設定
AioGetAiInputMethod	入力方式の取得
チャンネル	
AioGetAiMaxChannels	最大使用可能チャンネル数の取得
AioSetAiChannels	チャンネル数の設定
AioGetAiChannels	チャンネル数の取得
AioSetAiChannelSequence	変換チャンネル順序の設定
AioGetAiChannelSequence	変換チャンネル順序の取得
レンジ	
AioSetAiRange	指定チャンネルのレンジを設定
AioSetAiRangeAll	全チャンネルのレンジを設定
AioGetAiRange	指定チャンネルのレンジを取得
転送方式	
AioSetAiTransferMode	転送方式の設定
AioGetAiTransferMode	転送方式の取得
AioSetAiTransferData	ユーザーバッファの設定
AioSetAiAttachedData	添付データの設定
AioGetAiSamplingDataSize	1サンプリングあたりの転送サイズを取得

メモリ形式	
AioSetAiMemoryType	データ格納用メモリ形式の設定
AioGetAiMemoryType	データ格納用メモリ形式の取得
リピート	
AioSetAiRepeatTimes	リピート回数の設定
AioGetAiRepeatTimes	リピート回数の取得
クロック	
AioSetAiClockType	クロック種類の設定
AioGetAiClockType	クロック種類の取得
AioSetAiSamplingClock	変換速度の設定
AioGetAiSamplingClock	変換速度の取得
AioSetAiScanClock	チャンネルスキャンクロックの設定
AioGetAiScanClock	チャンネルスキャンクロックの取得
AioSetAiClockEdge	外部クロックの入力タイミングを設定
AioGetAiClockEdge	外部クロックの入力タイミングを取得
開始条件	
AioSetAiStartTrigger	変換開始条件の設定
AioGetAiStartTrigger	変換開始条件の取得
AioSetAiStartLevel	レベルトリガ開始条件の設定(バイナリ値)
AioGetAiStartLevel	レベルトリガ開始条件の取得(バイナリ値)
AioSetAiStartLevelEx	レベルトリガ開始条件の設定(電圧/電流)
AioGetAiStartLevelEx	レベルトリガ開始条件の取得(電圧/電流)
AioSetAiStartInRange	インレンジ比較開始条件の設定(バイナリ値)
AioGetAiStartInRange	インレンジ比較開始条件の取得(バイナリ値)
AioSetAiStartInRangeEx	インレンジ比較開始条件の設定(電圧/電流)
AioGetAiStartInRangeEx	インレンジ比較開始条件の取得(電圧/電流)
AioSetAiStartOutOfRange	アウトレンジ比較開始条件の設定(バイナリ値)
AioGetAiStartOutOfRange	アウトレンジ比較開始条件の取得(バイナリ値)
AioSetAiStartOutOfRangeEx	アウトレンジ比較開始条件の設定(電圧/電流)
AioGetAiStartOutOfRangeEx	アウトレンジ比較開始条件の取得(電圧/電流)
停止条件	
AioSetAiStopTrigger	変換停止条件の設定
AioGetAiStopTrigger	変換停止条件の取得
AioSetAiStopTimes	サンプリング回数の設定
AioGetAiStopTimes	サンプリング回数の取得
AioSetAiStopLevel	レベルトリガ停止条件の設定(バイナリ値)
AioGetAiStopLevel	レベルトリガ停止条件の取得(バイナリ値)
AioSetAiStopLevelEx	レベルトリガ停止条件の設定(電圧/電流)
AioGetAiStopLevelEx	レベルトリガ停止条件の取得(電圧/電流)
AioSetAiStopInRange	インレンジ比較停止条件の設定(バイナリ値)
AioGetAiStopInRange	インレンジ比較停止条件の取得(バイナリ値)
AioSetAiStopInRangeEx	インレンジ比較停止条件の設定(電圧/電流)
AioGetAiStopInRangeE	インレンジ比較停止条件の取得(電圧/電流)
AioSetAiStopOutOfRange	アウトレンジ比較停止条件の設定(バイナリ値)
AioGetAiStopOutOfRange	アウトレンジ比較停止条件の取得(バイナリ値)
AioSetAiStopOutOfRangeEx	アウトレンジ比較停止条件の設定(電圧/電流)
AioGetAiStopOutOfRangeEx	アウトレンジ比較停止条件の取得(電圧/電流)

遅延サンプリング	
AioSetAiStopDelayTimes	変換停止遅延回数の設定
AioGetAiStopDelayTimes	変換停止遅延回数の取得
イベント	
AioSetAiEvent	Windowメッセージ通知の設定
AioGetAiEvent	Windowメッセージ通知の取得
AioSetAiCallBackProc	コールバック関数の登録
AioSetAiEventSamplingTimes	指定サンプリング回数格納イベントの設定
AioGetAiEventSamplingTimes	指定サンプリング回数格納イベントの取得
AioSetAiEventTransferTimes	指定転送数毎イベントの設定
AioGetAiEventTransferTimes	指定転送数毎イベントの取得
動作	
AioStartAi	変換開始
AioStartAiSync	変換開始(同期)
AioStopAi	変換停止
状態	
AioGetAiStatus	ステータスの取得
AioGetAiSamplingCount	変換回数の取得
AioGetAiStopTriggerCount	停止条件成立時の変換回数を取得
AioGetAiTransferCount	転送データ数の取得
AioGetAiTransferLap	メモリ周回数の取得
AioGetAiStopTriggerTransferCount	停止条件成立時の転送回数を取得
AioGetAiRepeatCount	リピート回数の取得
データ取得	
AioGetAiSamplingData	データ格納用メモリからデータの読み込み(バイナリ値)
AioGetAiSamplingDataEx	データ格納用メモリからデータの読み込み(電圧/電流)
リセット	
AioResetAiStatus	ステータスのリセット
AioResetAiMemory	データ格納用メモリのリセット

③アナログ出力関数一覧

簡易出力	
AioSingleAo	指定チャンネルを1回出力(バイナリ値指定)
AioSingleAoEx	指定チャンネルを1回出力(電圧/電流値指定)
AioMultiAo	複数チャンネルを1回出力(バイナリ値指定)
AioMultiAoEx	複数チャンネルを1回出力(電圧/電流値指定)
分解能	
AioGetAoResolution	分解能を取得
チャンネル	
AioSetAoChannels	チャンネル数の設定
AioGetAoChannels	チャンネル数の取得
AioGetAoMaxChannels	最大使用可能チャンネル数の取得

レンジ	
AioSetAoRange	指定チャンネルのレンジを設定
AioSetAoRangeAll	全チャンネルのレンジを設定
AioGetAoRange	指定チャンネルのレンジを取得
転送方式	
AioSetAoTransferMode	転送方式の設定
AioGetAoTransferMode	転送方式の取得
AioSetAoTransferData	ユーザーバッファの設定
AioGetAoSamplingDataSize	1 サンプルあたりの転送サイズを取得
メモリ形式	
AioSetAoMemoryType	データ出力用メモリ形式の設定
AioGetAoMemoryType	データ出力用メモリ形式の取得
リピート	
AioSetAoRepeatTimes	リピート回数の設定
AioGetAoRepeatTimes	リピート回数の取得
クロック	
AioSetAoClockType	クロック種類の設定
AioGetAoClockType	クロック種類の取得
AioSetAoSamplingClock	変換速度の設定
AioGetAoSamplingClock	変換速度の取得
AioSetAoClockEdge	外部クロックの入力タイミングを設定
AioGetAoClockEdge	外部クロックの入力タイミングを取得
データ設定	
AioSetAoSamplingData	出力データ設定(バイナリ値指定)
AioSetAoSamplingDataEx	出力データ設定(電圧/電流値指定)
AioGetAoSamplingTimes	出力データ取得
開始条件	
AioSetAoStartTrigger	変換開始条件の設定
AioGetAoStartTrigger	変換開始条件の取得
停止条件	
AioSetAoStopTrigger	変換停止条件の設定
AioGetAoStopTrigger	変換停止条件の取得
イベント	
AioSetAoEvent	Windowメッセージ通知の設定
AioGetAoEvent	Windowメッセージ通知の取得
AioSetAoCallBackProc	コールバック関数の登録
AioSetAoEventSamplingTimes	指定サンプリング回数格納イベントの設定
AioGetAoEventSamplingTimes	指定サンプリング回数格納イベントの取得
AioSetAoEventTransferTimes	指定転送数毎イベントの設定
AioGetAoEventTransferTimes	指定転送数毎イベントの取得

動作	
AioStartAo	変換開始
AioStopAo	変換停止
AioEnableAo	指定チャンネルのリレー回路をONに設定
AioDisableAo	指定チャンネルのリレー回路をOFFに設定
状態	
AioGetAoStatus	ステータスの取得
AioGetAoSamplingCount	変換回数の取得
AioGetAoTransferCount	転送データ数の取得
AioGetAoTransferLap	メモリ周回数の取得
AioGetAoRepeatCount	リピート回数の取得
リセット	
AioResetAoStatus	ステータスのリセット
AioResetAoMemory	データ設定用メモリのリセット

④ デジタル入力関数一覧

フィルタ	
AioSetDiFilter	フィルタ設定
AioGetDiFilter	フィルタ設定値取得
簡易入力	
AioInputDiBit	指定ビットから1データ入力
AioInputDiByte	指定ポートから1バイト分のデータ入力

⑤ デジタル出力関数一覧

簡易入力	
AioInputDiBit	指定ビットから1データ入力
AioInputDiByte	指定ポートから1バイト分のデータ入力

⑥ カウンタ用関数一覧

チャンネル	
AioGetCntMaxChannels	最大使用可能チャンネル数の取得
動作条件	
AioSetCntComparisonMode	比較カウント一致発生時のカウンタ動作設定
AioGetCntComparisonMode	比較カウント一致発生時のカウンタ動作取得
プリセット	
AioSetCntPresetReg	プリセットロード値の設定
比較カウント	
AioSetCntComparisonReg	比較カウントロード値の設定
クロック	
AioSetCntInputSignal	クロック種類の設定
AioGetCntInputSignal	クロック種類の取得

イベント	
AioSetCntEvent	Windowメッセージ通知の設定
AioGetCntEvent	Windowメッセージ通知の取得
AioSetCntCallBackProc	コールバック関数の登録
フィルタ	
AioSetCntFilter	フィルタ設定
AioGetCntFilter	フィルタ設定値取得
動作	
AioStartCnt	動作開始
AioStopCnt	動作停止
AioPresetCnt	カウンタのプリセット
状態	
AioGetCntStatus	ステータスの取得
AioGetCntCount	現在のカウンタ値を取得
リセット	
AioResetCntStatus	ステータスのリセット

⑦タイマ関数一覧

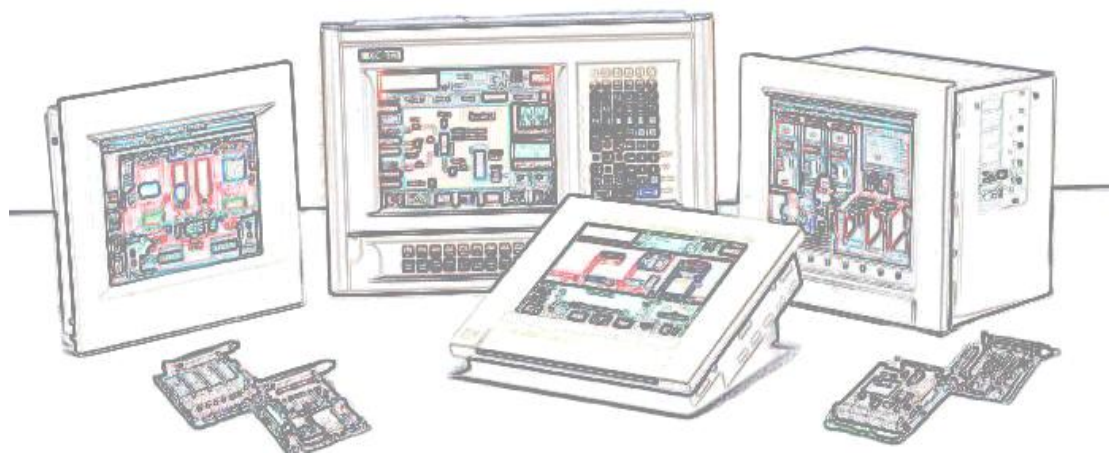
インターバルタイマ	
AioSetTmEvent	タイマイベントの設定
AioGetTmEvent	タイマイベントの取得
AioSetTmCallBackProc	コールバックルーチンの登録
AioStartTmTimer	インターバルタイマを開始
AioStopTmTimer	インターバルタイマを停止
経過時間測定	
AioStartTmCount	経過時間の測定を開始
AioStopTmCount	経過時間を取得し測定を停止
AioLapTmCount	現在の経過時間を取得
AioResetTmCount	経過時間をリセット
ウェイト	
AioTmWait	現在のスレッドの実行を指定された間隔だけ中断

⑧イベントコントローラ専用関数一覧

イベントコントローラ	
AioSetEcuSignal	イベントコントローラの信号設定
AioGetEcuSignal	イベントコントローラの信号設定値取得

第5章

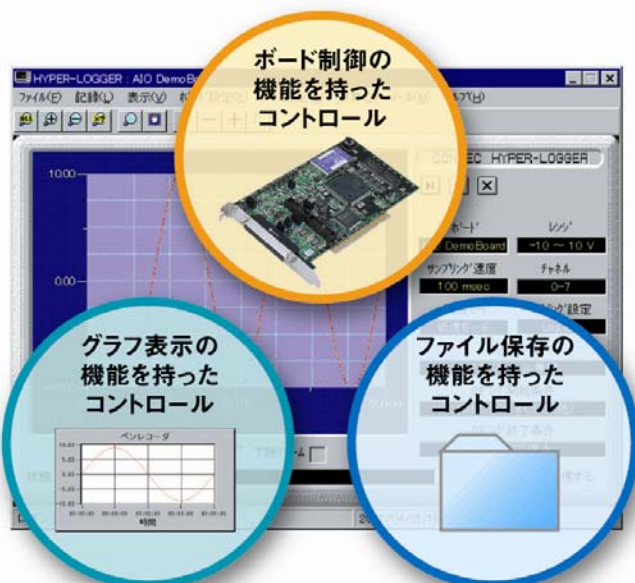
ActiveXによる コンポーネントプログラミング



第5章 ActiveXによるコンポーネントプログラミング

5-1. コンポーネントプログラミングとは

現在、アプリケーションのプログラミング手法は、よりオブジェクト指向を意識した『コンポーネントプログラミング』が主流になってきています。例えば、周辺機器からデータを入力し、グラフ表示およびファイル保存する計測アプリケーションを作成する場合を考えてみましょう。従来のプログラミング手法では、すべての機能を一からプログラミングし、アプリケーションを作成していました。そのため、多くの開発工数がかかり、また高度なスキルも要求されていました。それに対して、コンポーネントプログラミングは、特定の機能を持ったソフトウェア部品（コントロール）を組み合わせ、アプリケーションを作成します。この手法を用いれば、コントロールが既に持っている機能に関する部分のプログラミングは必要ありませんので、開発工数の削減はもちろん、高度なプログラミングスキルも必要とせず、高機能なアプリケーションが作成できるのです。



ボード制御の機能を持ったコントロール

グラフ表示の機能を持ったコントロール

ファイル保存の機能を持ったコントロール

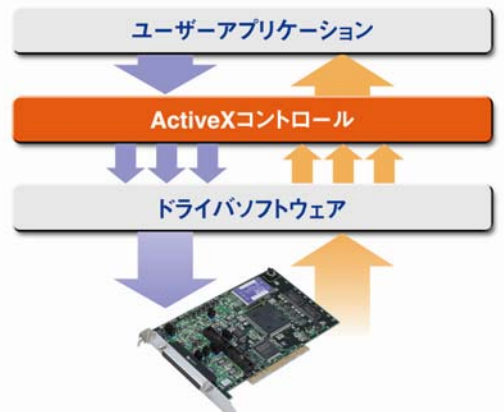
様々な機能を持ったソフトウェア部品（コントロール）を組み合わせ、高度なアプリケーションを短期間で開発可能なのが『コンポーネントプログラミング』の一番のメリット。

5-2. ActiveXコントロールとは

コンポーネントプログラミングで、現在もっとも多く利用されているのが、ActiveXコントロールと呼ばれるものです。ActiveXコントロールとはマイクロソフト社がプログラムを部品単位で構成し、再利用するための方法として提唱している、32ビットWindows環境用ソフトウェア部品の総称で、これを利用することで様々なメリットを得ることができます。

- ①省プログラミング（開発効率・生産性の向上）。
- ②ActiveXをサポートしている開発言語（コンテナ）で共通使用可能。
- ③動作検証されたコントロールを使用することでデバッグ工数削減。
- ④必要な機能だけを使用した柔軟性の高いアプリケーションが作成可能。

弊社では、もっと簡単に計測・制御システムを開発したい！
というご要望にお答えするため、各種ハードウェア制御、グラフ表示をはじめとする画面表示、ファイル保存、解析などシステム開発に必要なActiveXコントロールをワンパッケージ化した『ACX-PAC (W32)』を発売しています。なかでも各種ハードウェア制御用コントロールは、ドライバソフトウェアをコントロール化、ボード/カードの操作に必要な機能をあらかじめ持たせ、より簡単にハードウェアを操作できるようになっているコントロールです。各種設定をプロパティページに集約することにより、大幅な『省プログラミング』を実現します。



5-3.計測システム開発用ActiveXコンポーネント集『ACX-PAC (W32)』紹介

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。『実例集』の紹介は、巻末の付録を参照してください。

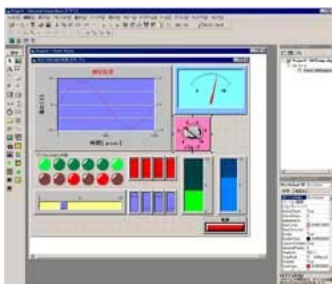
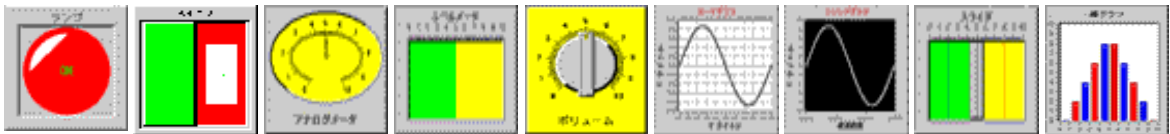
画面作成用コンポーネント	
X-Yグラフ	1次元、または2次元配列データのグラフを同時に32ライン表示。マウスによる拡大・縮小・カーソル・移動やサブ軸などの機能
棒グラフ	ヒストグラム表示に最適な棒グラフを表示。
トレンドグラフ	スクロールしていくグラフを8ラインまで表示。モニタリングに最適。
ランプ	デジタル入力状態の表示に最適なランプを表示。
スイッチ	デジタル出力や様々な設定のON/OFF等の表示に最適なスイッチ。
アナログメータ	取得データをアナログメータで表示。
レベルメータ	取得データをレベルメータで表示。
スライダ	データ設定に最適なスライダスイッチ。
ボリューム	データ設定に最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。
GPIO通信	弊社製GPIO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	対応開発環境	対応パソコン
<ul style="list-style-type: none"> •Windows Vista Ultimate •Windows Vista Enterprise •Windows Vista Business •Windows Vista Home Premium および Basic •Microsoft Windows XP Professional •Microsoft Windows XP Home Edition •Microsoft Windows Server 2003 •Microsoft Windows 2000 Professional •Microsoft Windows NT Ver. 4.0 (SP3 以上) <ul style="list-style-type: none"> + Internet Explorer 4.01 以上 •Microsoft Windows Me •Microsoft Windows 98 およびSecond Edition •Microsoft Windows 95 (SP1 以上) <ul style="list-style-type: none"> + Internet Explorer 4.01 以上 詳しくは、弊社ホームページをご確認ください。	<ul style="list-style-type: none"> •Microsoft Visual Basic Ver.6.0, 5.0 •Microsoft Visual C++ Ver.6.0, 5.0 •Microsoft Visual Basic 2005, .NET 2003, .NET 2002 •Microsoft Visual C++ 2005, .NET 2003, .NET 2002 •Microsoft Visual C# 2005, .NET 2003, .NET 2002 •Microsoft Excel 2003 (VBA 6.4), 2002 (VBA 6.3), 2000 (VBA 6.0), 97 (VBA 5.0) •Borland Delphi Ver.7, Ver.5, Ver.4 •National Instruments LabVIEW 8.20, 8, 7.1, 7.0, 6.1, 6i 詳しくは、弊社ホームページをご確認ください。	<ul style="list-style-type: none"> • IBM PC/AT 互換機、DOS/V 機 その他 <ul style="list-style-type: none"> • Pentium100MHz以上のCPUを推奨 • プログラミング言語（コンテナ）が正常に動作する環境

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！

5-4. アナログ入出力カードのFIFOメモリを使用した『簡易オシロスコープ』の作成

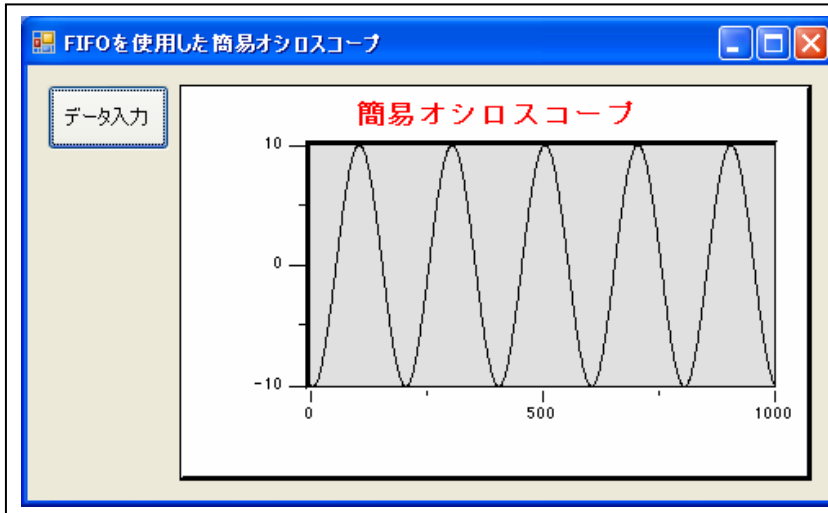
第4章『Visual Basicによるアナログ入出力プログラミング』の【4-6】で作成した、『FIFOメモリを使用した高速サンプリング(簡易オシロスコープ:波形表示)』プログラムを、ACX-PAC(W32)を実際に使用して再現してみましょう。複雑なプログラムが如何に簡単に作成できるかが体験できると思います。なお、これから使用するACX-PAC(W32) Ver.4.11は、体験版(無償)を用意しており、弊社ホームページ上から簡単に請求できます。体験版のインストール方法などは、体験版の解説書を参照してください。なお、ここからの説明は、ACX-PAC(W32) Ver.4.11が既にパソコンにセットアップ(インストール)されているものとして進めていきます。

◎ 計測システム開発用ActiveXコンポーネント集『ACX-PAC(W32)』スペシャルサイト

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

5-4-1.プログラム概要

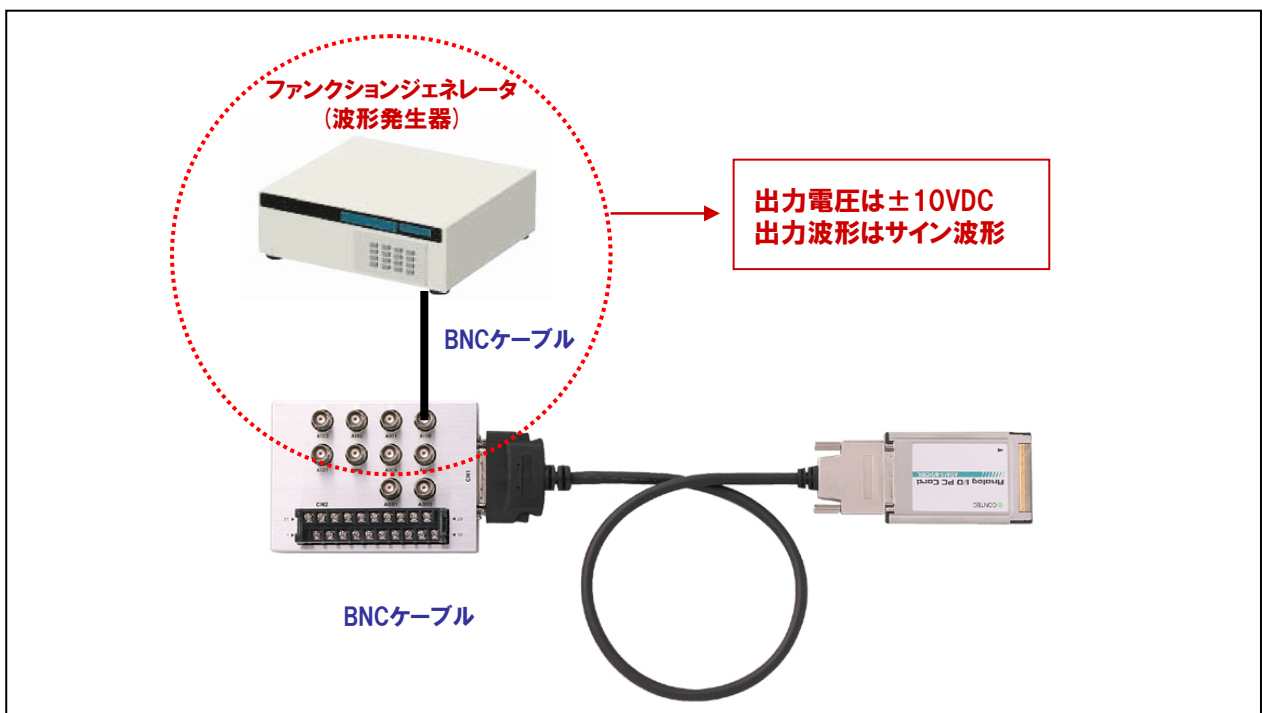
バッファメモリを使用して、1000 μ sec周期のデータを1000回サンプリングし、波形表示するプログラムです。アナログ入出力カード『ADA16-8/2(CB)L』の『アナログ入力0ch』のみ使用します。



実行画面イメージ

5-4-2.実行環境

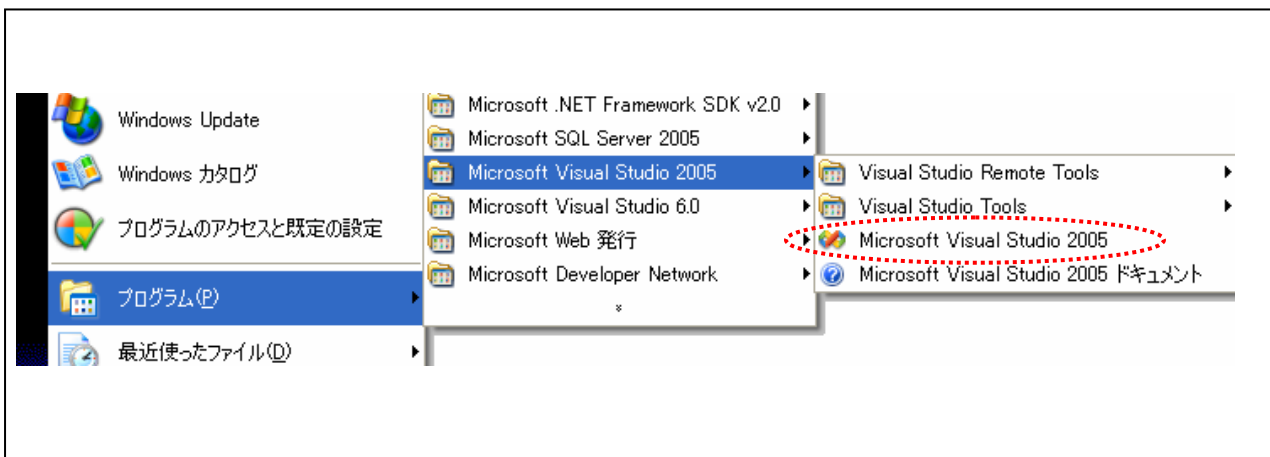
【3-6-3.実習環境の構築】の環境を使用します。ファンクションジェネレータは、次の設定を事前に行っています。出力アナログ信号は ± 10 VDC (VDC:直流を示す)、出力波形はサイン波としています。



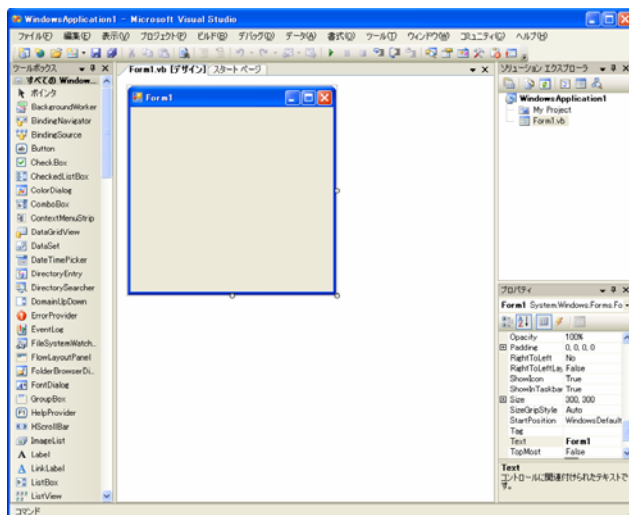
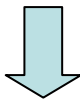
5-4-3.プログラム作成手順① (Visual Basicの起動)

まず最初にVisual Basic2005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』 - 『プログラム』 - 『Microsoft Visual Studio2005』 - 『Microsoft Visual Studio2005』を選択します。



メニューの『ファイル』 - 『新しいプロジェクト』を選択します。『新しいプロジェクト』ダイアログが表示されますので、『プロジェクトの種類』から『Visual Basic』 - 『Windows』を選択し、『テンプレート』から『Windows アプリケーション』を選択して、『OK』ボタンをクリックしてください。



5-4-4.プログラム作成手順② (ActiveXコントロールの追加と各コントロールの貼り付け)

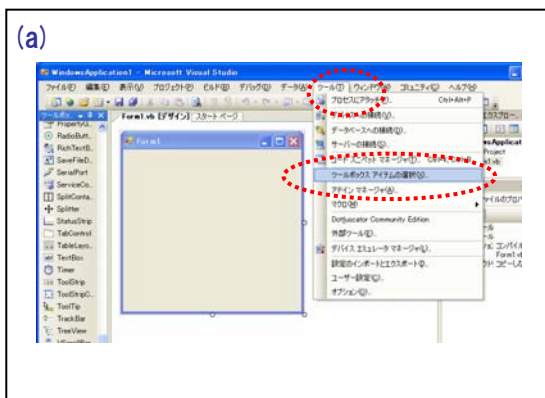
プロジェクトへ『ACX-PAC (W32) Ver.4.0』が提供するActiveXコントロールの追加を行います。

これらのコントロールは、標準の状態ではコントロールツールボックスに登録されていないため、以下の手順で、本プロジェクトに『CONTEC ACX Analog Control』および『CONTEC ACX X-Y Graph Control』を追加する処理を行います。

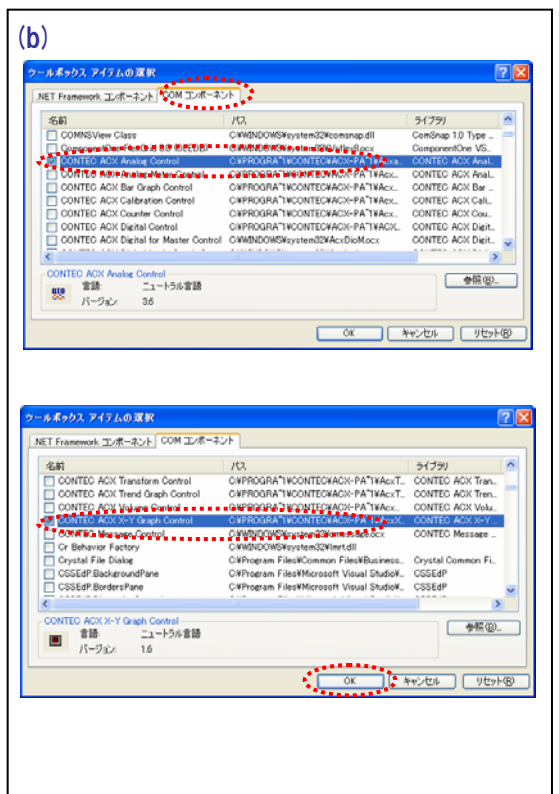
- Visual Basic2005のメニュー『ツール』 - 『ツールボックス アイテムの選択』を選択します。
- Visual Basic2005で使用できるツールの一覧が表示されますので、この中から、『COM コンポーネント』タブを選択し、『CONTEC ACX Analog Control』および、『CONTEC ACX X-Y Graph Control』のチェックボックスにチェックをつけ、『OK』をクリックします。
- コントロールツールボックスに『CONTEC ACX Analog Control』および、『CONTEC ACX X-Y Graph Control』が追加されます。
- 『CONTEC ACX Analog Control』および、『CONTEC ACX X-Y Graph Control』を選択 (クリック) して、フォームに貼り付けます。

なお、『CONTEC ACX Analog Control』は、プログラム実行時には不可視となりますが、適当な場所に貼り付け可能です。コマンドボタンも同時に貼り付けます

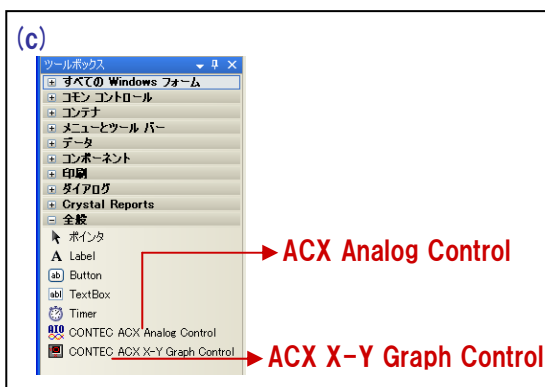
(a)



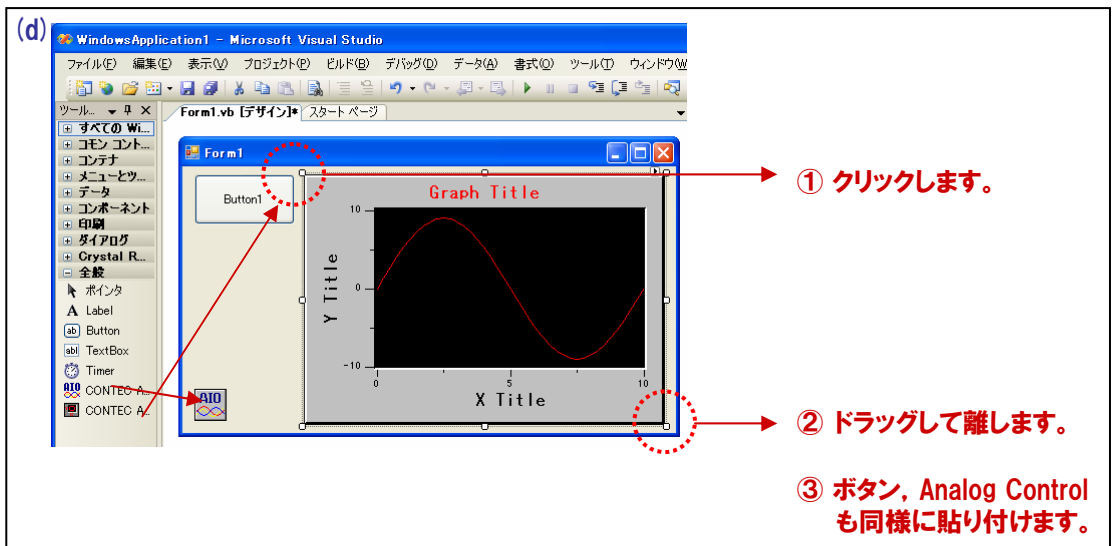
(b)



(c)



(d)



① クリックします。

② ドラッグして離します。

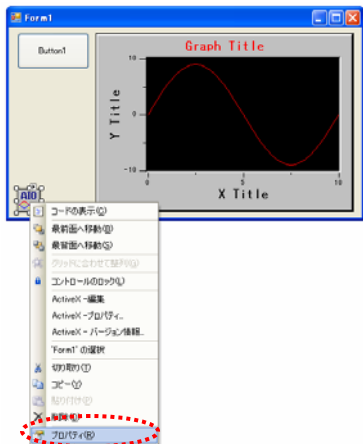
③ ボタン, Analog Control も同様に貼り付けます。

5-4-5.プログラム作成手順③ (各コントロールのプロパティ設定)

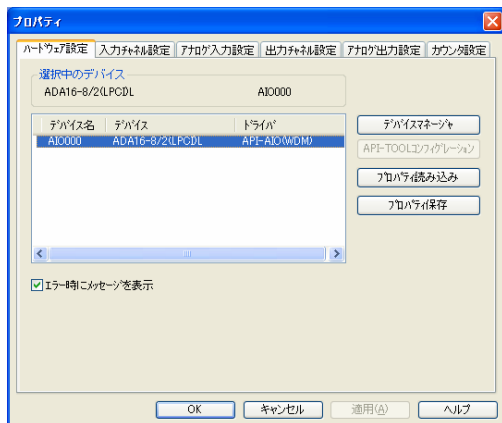
① CONTEC ACX Analog Control

ACX Analog Controlは弊社製アナログ入出力ボード/カード/USBモジュールを、Visual Basicなどから簡単に使用することができるActiveXコンポーネントです。複雑なプログラミングをすることなく、実装された『プロパティページ』によって、アナログ入出力の各種設定が可能です。すべての設定終了後は『OKボタン』で閉じます。

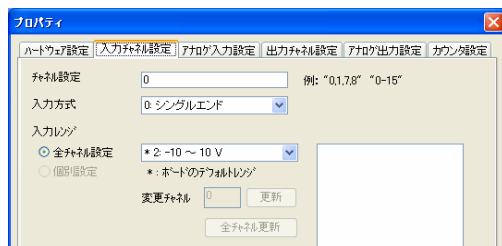
ACX Analog Controlを選択した状態で、右クリックします。メニューの中から『プロパティ』を選択します。実装されているプロパティページが自動で開かれます。



プロパティページが開かれます。ハードウェア設定ページでは、コントロールが制御するハードウェアを選択します。ハードウェアは登録されていれば自動で取得し、一覧に表示を行います。今回は、使用している『ADA12-8/2 (CB) L』を選択します。



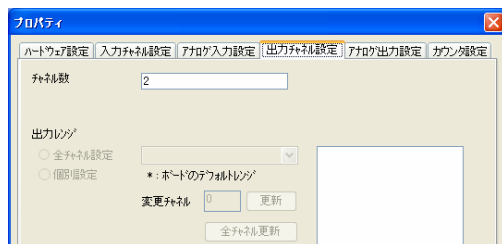
入力チャンネル設定ページでは、入力チャンネル数、入力方式、入力レンジの設定が行えます。今回は、0チャンネルのみ使用しますので、チャンネル設定に"0"を設定します。入力方式はシングルエンド、入力レンジは-10~10Vに設定します。



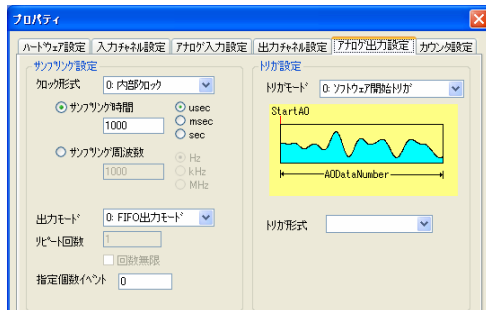
アナログ入力設定ページでは、クロックは"内部クロック"、サンプリング時間は"1000 μ sec"、サンプリング回数は、"1000"、トリガモードは、"ソフトウェア開始トリガ"に設定します。



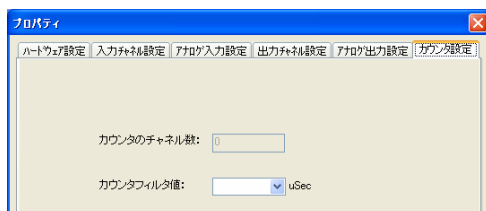
今回は使用しませんが、アナログ出力についても簡単に設定することができます。出力チャンネル設定ページでは、出力チャンネル数、出力レンジの設定が行えます。



今回は使用しませんが、アナログ出力設定では、クロック形式、サンプリングに関する設定、トリガに関する設定が行えます。



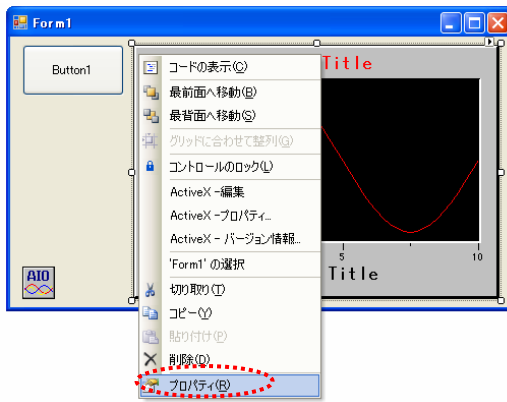
今回は使用しませんが、カウンタについても簡単に設定することができます。カウンタ設定ページでは、カウンタのチャンネル数、カウンタのフィルタ値の設定が行えます。



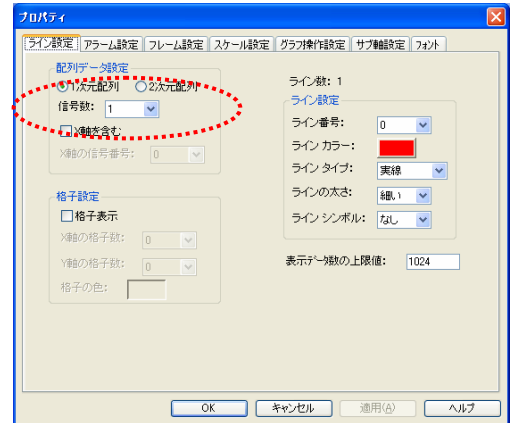
② CONTEC ACX X-Y Graph Control

ACX X-Y Graph Controlは、X-Y配列（1・2次元）データのグラフを表示する機能を持ったコントロールです。複雑なプログラミングをすることなく、実装された『プロパティページ』によって、様々なグラフの要素やプロパティの各種設定が可能です。すべての設定終了後は『OKボタン』で閉じます。

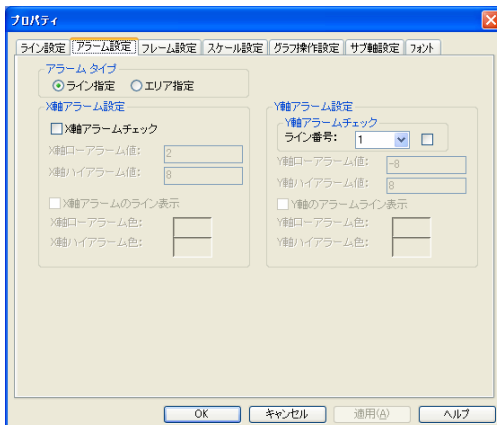
ACX X-Y Graph Controlを選択した状態で、右クリックします。メニューの中から『プロパティ』を選択します。実装されているプロパティページが自動で開かれます。



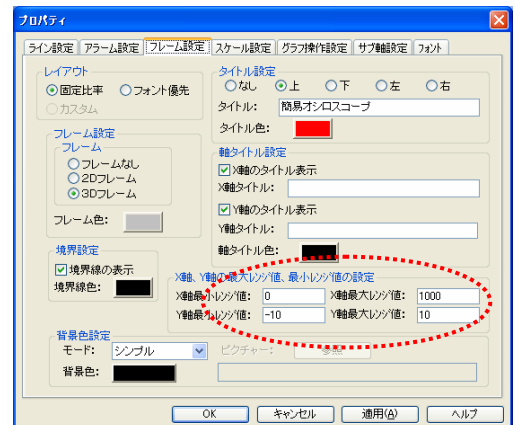
プロパティページが開かれます。ライン設定ページでは、配列データの指定や信号数（最大32ライン）を行います。今回は”1次元配列”で、信号数は”1”を設定します。ラインの色や太さ、タイプなどもこのページで設定できます。



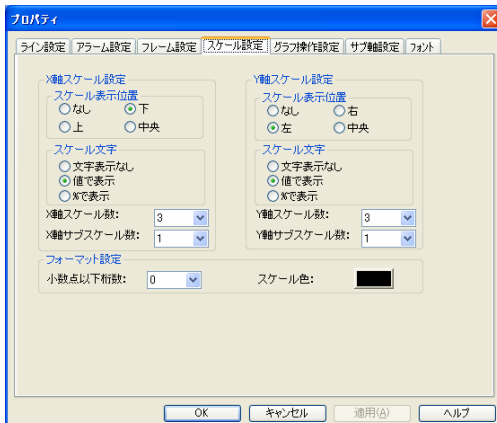
ACX X-Y Graph Controlはアラームの機能を有しています。アラーム設定ページでは、データの上限值と下限値の設定などが行えます。今回はアラーム機能は使用しませんので、設定は行いません。



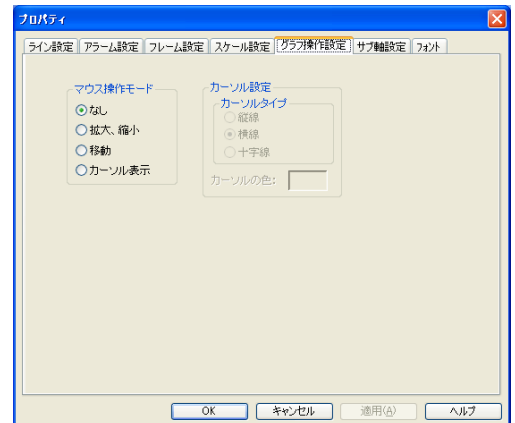
フレーム設定ページでは、主にグラフの外観の設定を行います。外観に関しては好みの外観に合わせて設定を行ってください。レンジに関しては、Y軸は-10~+10、X軸は0から1000の設定を行ってください。



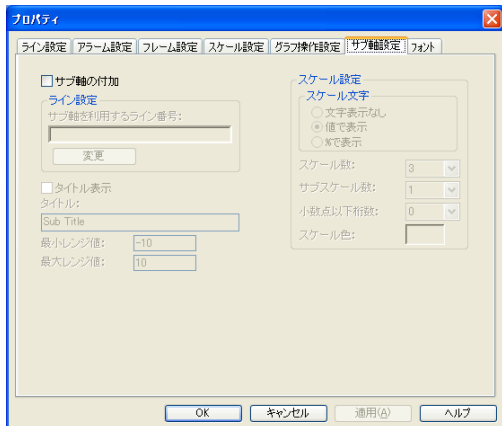
スケール設定ページでは、X軸・Y軸それぞれのスケールに関する設定が可能です。今回は、デフォルトのままの設定です。



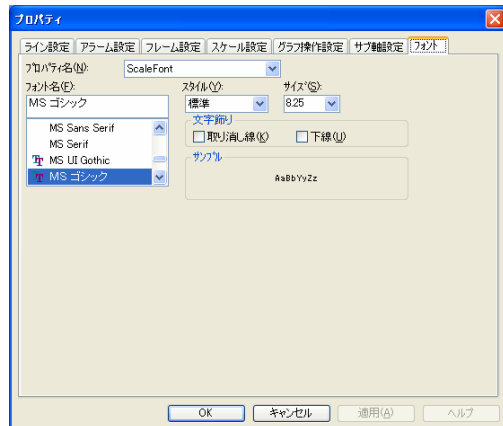
グラフ操作設定ページでは、グラフを操作する際の各種設定が可能です。今回は使用しません。



サブ軸設定ページでは、グラフにサブ軸を付加するかしないか、スケールの表示をするかなどの設定が可能です。今回は使用しません。



フォント設定ページでは、各タイトルのフォント設定、スケールのフォント設定が行えます。今回は、好みの設定を行ってください。



③ コマンドボタンのプロパティ設定と各コントロールのオブジェクト名の設定

最後に、コマンドボタンコントロールのプロパティ設定と、各オブジェクトのオブジェクト名の設定（確認）を行います。これで、画面の準備はすべて完了です。

■ Form
◎Text = 『FIFOを使用した簡易オシロスコープ』

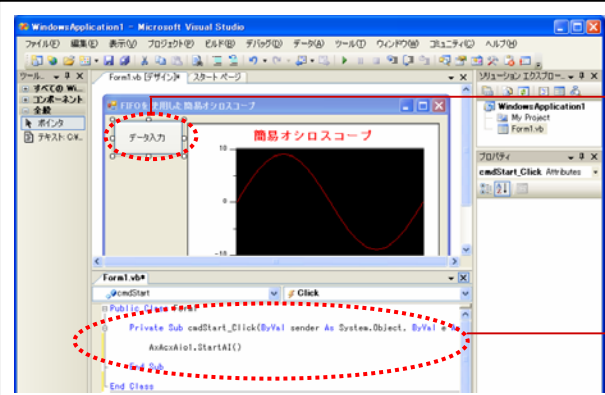
■ ACX X-Y Graphコントロール
◎Name = 『AxAcxXY1』 →デフォルト値

■ ACX Analog Graphコントロール
◎Name = 『AxAcxAio1』 →デフォルト値

■ データ入力ボタン
◎Text = 『データ入力』
◎Name = 『cmdStart』

5-4-6.プログラム作成手順④ (コードの記述)

データの入力処理 (A/D変換)とグラフ描画を行うためのコードを記述します。



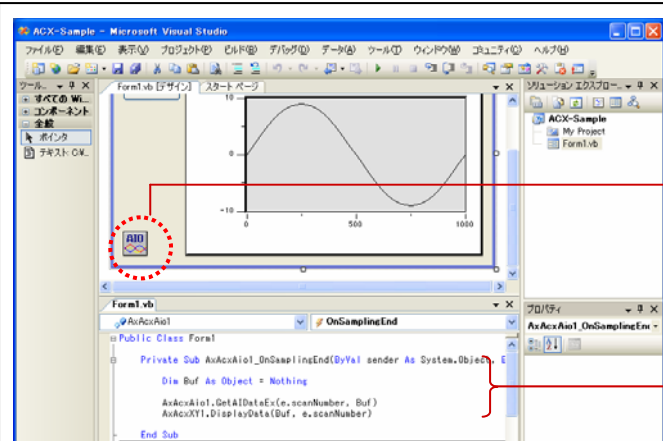
① 『データ入力』ボタンをダブルクリックします。

② コードウィンドウが開きます。
Private Sub cmdStart_Click () ~
End Subの間に下記のコードを記述します。

```
AxAcxAio1.StartAI ()
```

'変換開始

変換開始命令が出されると『ACX Analog Control』は、プロパティページで設定した条件に基づいてA/D変換を開始します。そして、設定された終了条件が満たされるとイベントを発生させます。今回は、サンプリングが1000回に達した時点でイベント『OnSamplingEnd』が発生します。このタイミングで、バッファメモリからのデータ取得と『ACX X-Y Graph Control』へのグラフ描画を行います。



① 『ACX Analog Control』をダブルクリックします。

② Private Sub AxAcxAio1_OnSamplingEnd () から、End Subの間にデータ取得処理とグラフ表示処理を記述します。

下記のコードを『Private Sub AxAcxAio1_OnSamplingEnd (ByVal sender As System.Object, ByVal e As

AxAcxAioLib._DACxaioEvents_OnSamplingEndEvent) Handles AxAcxAio1.OnSamplingEnd』から『End Sub』の間に記述してください。

```
Dim Buffer As Object = Nothing
```

'データ格納用変数

```
AxAcxAio1.GetAIDataEx (e.scanNumber, Buffer)
```

'データを取得し、変数へ格納

```
AxAcxAio1.DisplayData (Buffer, e.ScanNumber)
```

'取得したデータをグラフに表示

解説: サンプリングが終了すると (今回は1000回)、OnSamplingEndイベントが発生します。その際に引数として ScanNumberとChannelNumberの二つの情報がコントロールから渡されます。ScanNumberは、プロパティページで設定したサンプリング回数 (メモリ内にあるデータ数) が、ChannelNumberは、チャンネル数が渡されてきます (今回はこの情報は使用しません)。その情報を元に、データの取得とグラフ表示を行っています。

アナログ入力開始関数 『StartAI』 リファレンス (ACX Analog Control)

- 機能 プロパティページで設定された条件でアナログ入力を開始します。
- 書式 Visual Basic2005の場合

```
オブジェクト名.StartAI ()
```

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、割愛しています。
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。

データ取得処理関数 『GetAIDataEX』 リファレンス (ACX Analog Control)

- 機能 デバイスメモリからデータを取得します(バイナリ)。
- 書式 Visual Basic2005の場合

```
Dim ScanNumber As Integer  
Dim Buffer As Object  
オブジェクト名.GetAIDataEx (ScanNumber, Buffer)
```

- 引数 ScanNumber: 読み込むサンプリング回数を設定します。

Buffer :データを格納する配列を指定します。

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、割愛しています。
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。

データ表示処理関数 『DisplayData』 リファレンス (ACX X-Y Graph Control)

- 機能 複数データを一括して、X-Yグラフにデータを表示します。。
- 書式 Visual Basic2005の場合

```
Dim DataXY As Single  
Dim DataPointNumber As Integer  
オブジェクト名.DisplayData (DataXY, DataPointNumber)
```

- 引数 DataXY : X軸およびY軸のデータが格納されている配列を指定します。

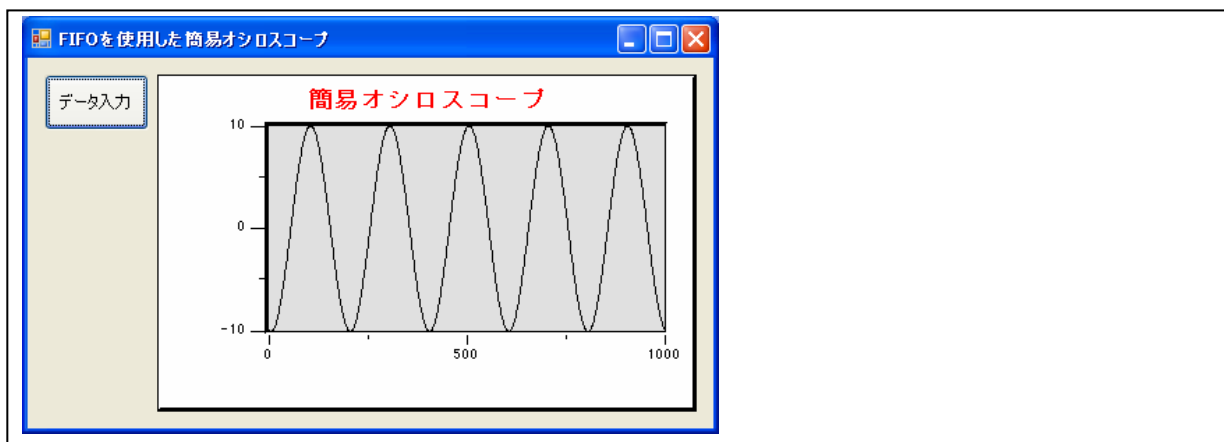
DataPointNumber :データ表示するデータポイント数を指定します。

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、割愛しています。
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。

5-4-7.プログラムの実行

『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、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『データ入力』をクリックして実行してみましょう。1000 μ sec 間隔のデータが1000回サンプリングされ瞬時にグラフに波形が表示されました。



5-4-8.ActiveXコントロールを使用した簡易オシロスコープ プログラムリスト

```
Private Sub cmdStart_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdStart.Click
```

```
    AxAcxAio1.StartAI ()                '変換開始
```

```
End Sub
```

```
Private Sub AxAcxAio1_OnSamplingEnd (ByVal sender As System.Object, ByVal e As AxACXAIOLib._DacxaioEvents_OnSamplingEndEvent) Handles AxAcxAio1.OnSamplingEnd
```

```
    Dim Buffer As Object = Nothing        'データ格納用変数  
    AxAcxAio1.GetAIData (e.scanNumber, Buffer) 'データを取得し、変数へ格納  
    AxAcxXY1.DisplayData (Buffer, ScanNumber) '取得したデータをグラフに表示
```

```
End Sub
```

【4-5-15. FIFOメモリを使用した高速サンプリング(簡易オシロスコープ:波形表示)リスト】と比較してみてください。その違いはあきらかです。このように弊社製アナログ入出力デバイスを制御する機能を持ったソフトウェア部品(コントロール)と、グラフ表示の機能を持ったコントロールを組み合わせる事により、工数の削減だけでなく、より高度なアプリケーションを簡単に作り上げる事ができます。是非、計測システム開発用ActiveXコンポーネント集 ACX-PAC (W32) で、それを体感してください。

計測システム開発用ActiveXコンポーネント集『ACX-PAC (W32)』スペシャルサイト

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

第6章

リモート/0



第6章 リモートI/O

第4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が、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。

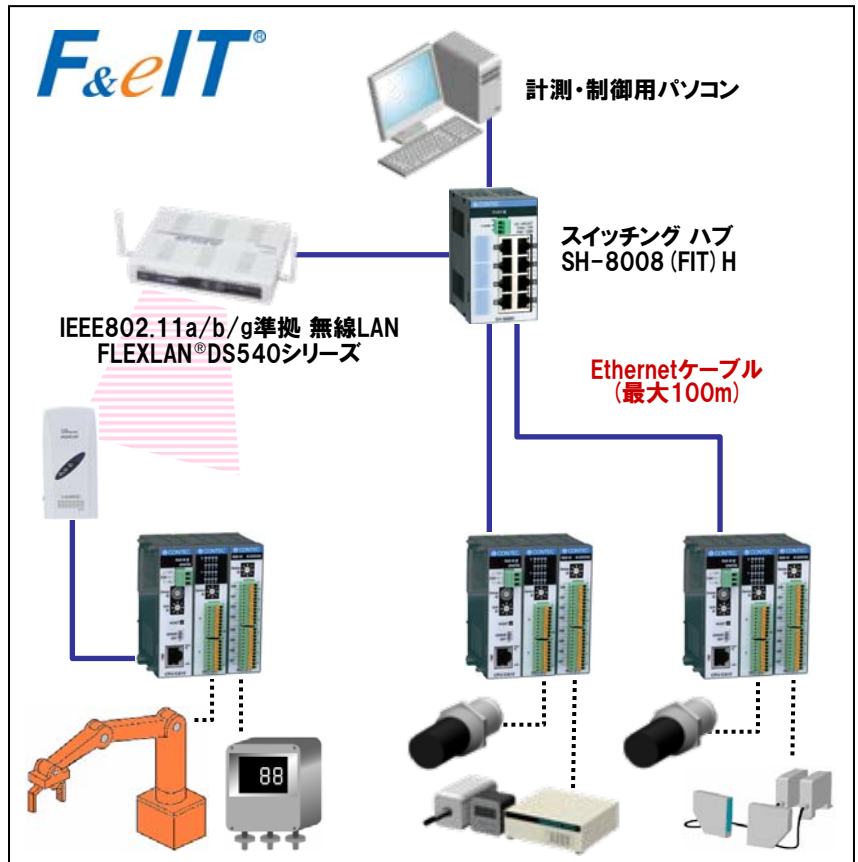


図6-1. 弊社のイーサネットベースリモートI/O製品を使用したシステム例

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(ほぼ名刺サイズ)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PCI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
絶縁型デジタル入出力モジュール	
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
非絶縁型デジタル入出力モジュール	
TTL (5VDC) 入出力8点	DIO-8D (FIT) GY
絶縁型デジタル入力モジュール	
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
絶縁型デジタル出力モジュール	
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
絶縁型アナログ入力モジュール	
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
絶縁型アナログ出力モジュール	
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
Pt100温度センサ入力モジュール	
Pt100温度センサ入力 4チャンネル	PTI-4 (FIT) GY
絶縁型カウンタモジュール	
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
リードリレー-接点出力モジュール	
125VAC/30VDC 2A リードリレー-接点出力 4点	RRY-4 (FIT) GY
シリアル コミュニケーションモジュール	
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 ※
GPIO コミュニケーションモジュール	
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。

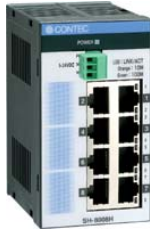


F&eITシリーズ スペシャルサイト <http://www.contec.co.jp/fit/>

④ F&eITシリーズ スwitchingHUB SH-8008 (FIT) H

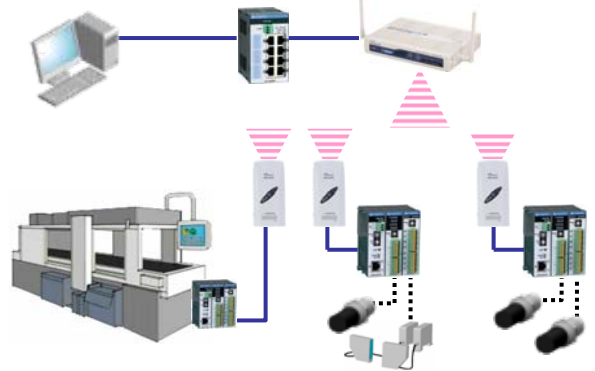
組み込みに便利な超小型・軽量の汎用Switching HUBです。外形幅はわずか52.4mmのコンパクト設計のため、一般的なHUBではスペース上で設置が困難な盤内、装置内にも設置可能です。電源仕様は5~24VDCのワイド入力に対応しているため、専用の外部電源ユニットを使用せずに、装置内の電源を使用することもできるなど組み込み用途に適した仕様となっています。

- 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。



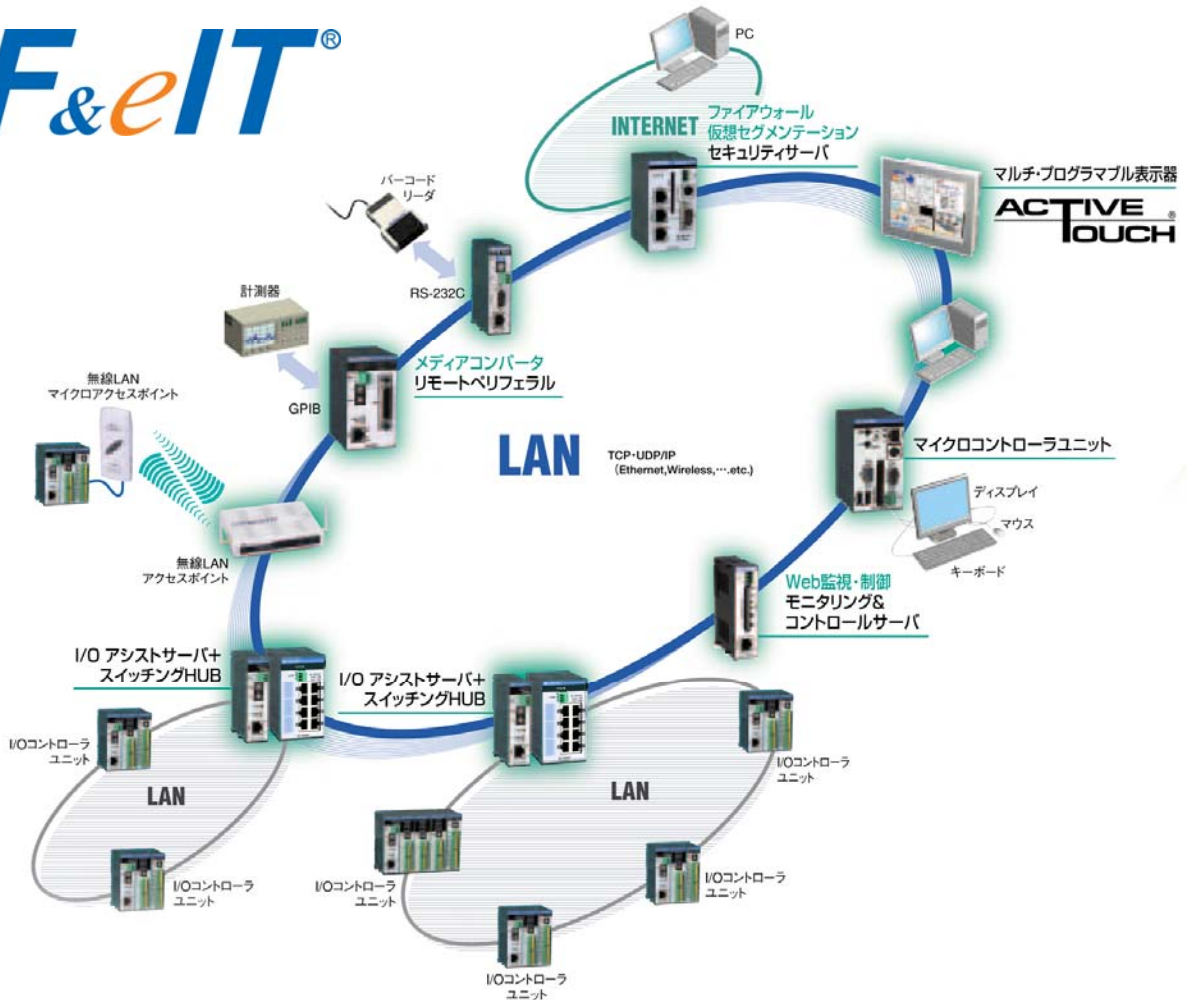
⑤ 無線LANアクセスポイントとステーション FLEXLAN DS540シリーズ

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®



F&eIT®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。

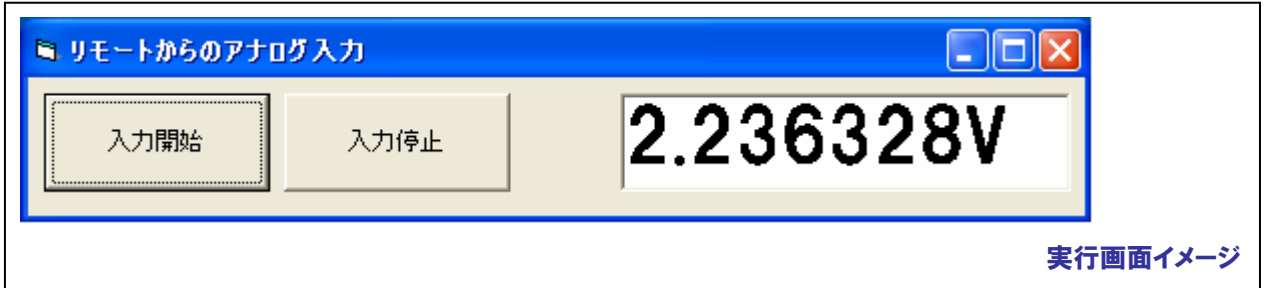
F&eIT®シリーズ スペシャルサイト
<http://www.contec.co.jp/fit/>

6-3. リモートI/Oによるアナログ入力プログラミング

弊社F&eITシリーズを使用したアナログ入力リモート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事がお分かりいただけると思います。

6-3-1. リモートI/Oによるアナログ入力プログラム概要

遠隔地にあるセンサ（本書ではセンサの代わりにファンクションジェネレータを使用）からのアナログ信号（電圧値）を入力して画面表示を行うプログラムを作成します。



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



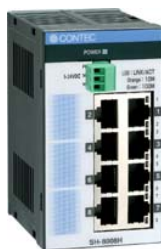
③ I/Oデバイスモジュール 絶縁型アナログ入力モジュール ADI12-8 (FIT) GY (分解能: 12ビット、差動入力8チャンネル)



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



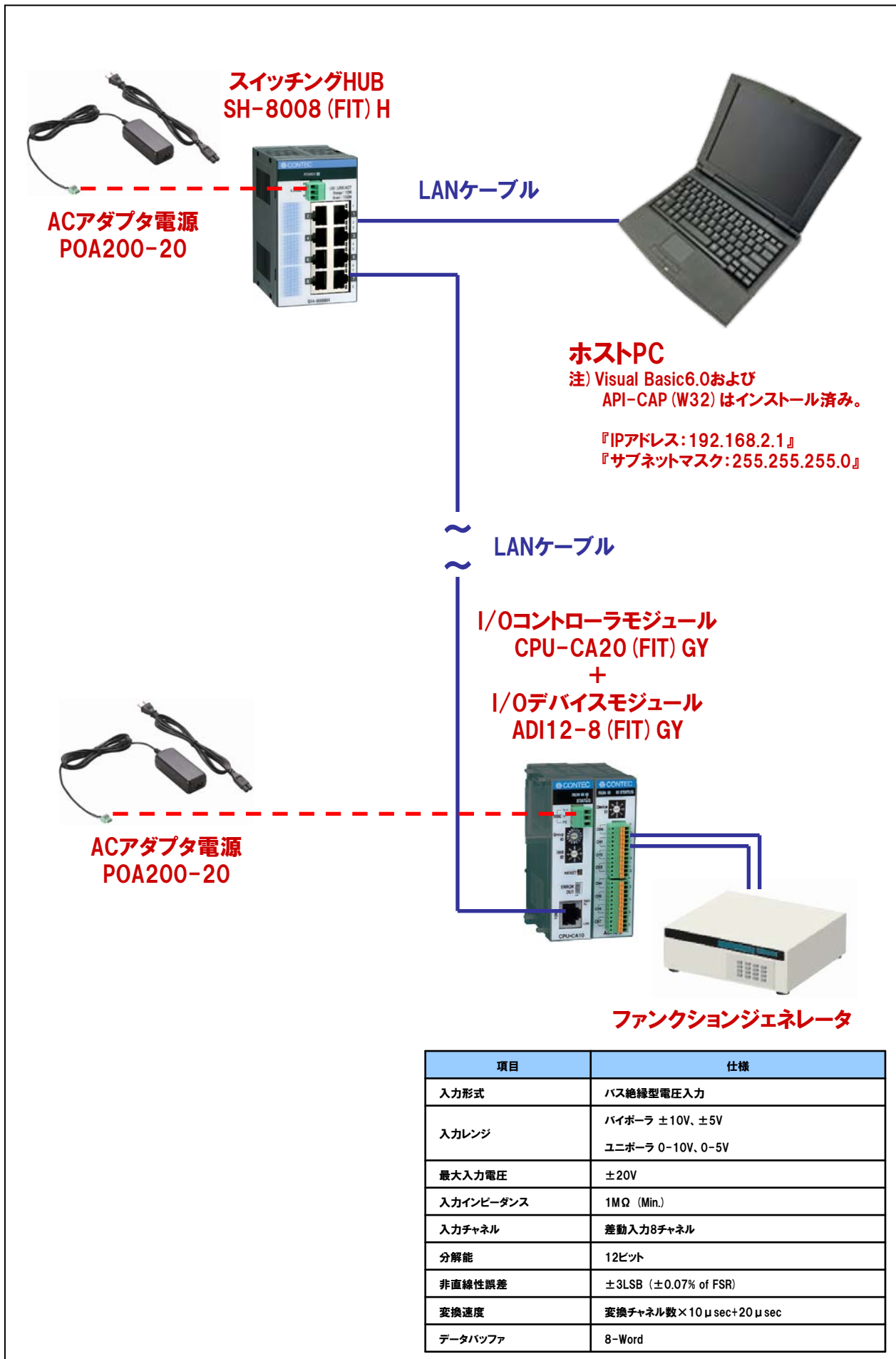
⑤ スイッチングHUB 組み込み型10M/100M自動認識スイッチングHUB SH-8008 (FIT) H

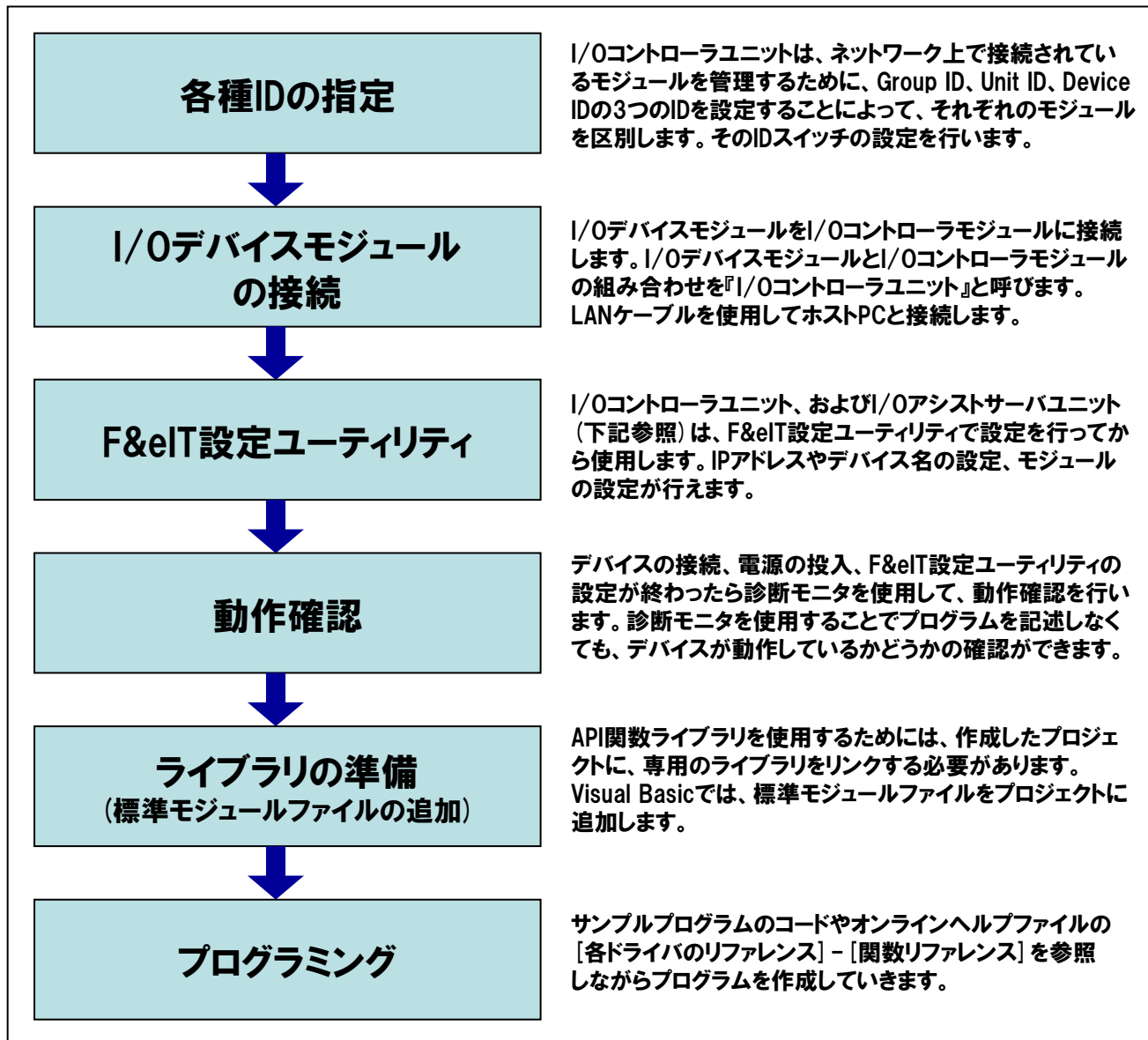


⑥ その他 (1) LANケーブル×2本 (2) 接続ケーブル (モジュール⇄ファンクションジェネレータ) (3) ファンクションジェネレータ (波形発生器)



6-3-3.機器接続図

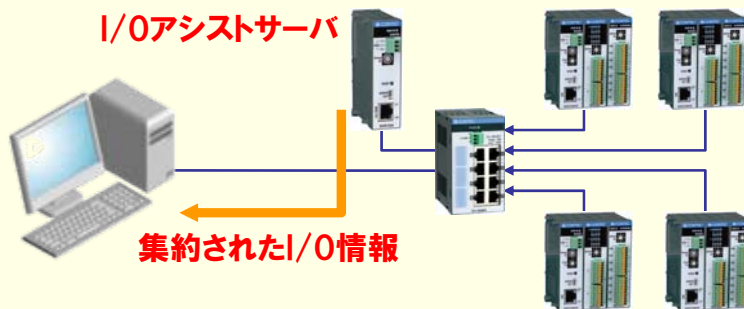




TOPICS

『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。

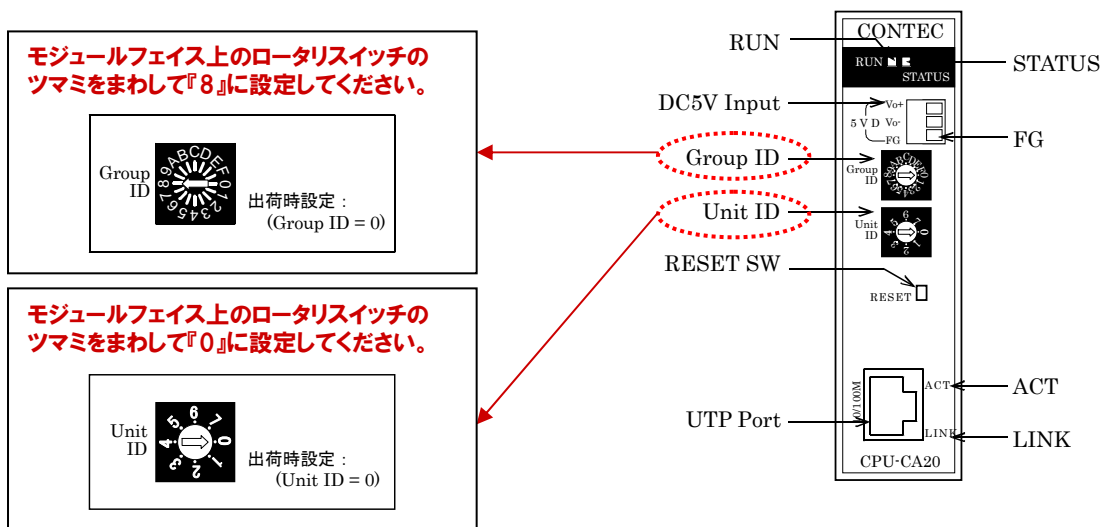
① I/Oコントローラモジュール CPU-CA20 (FIT) GYの設定

■ 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

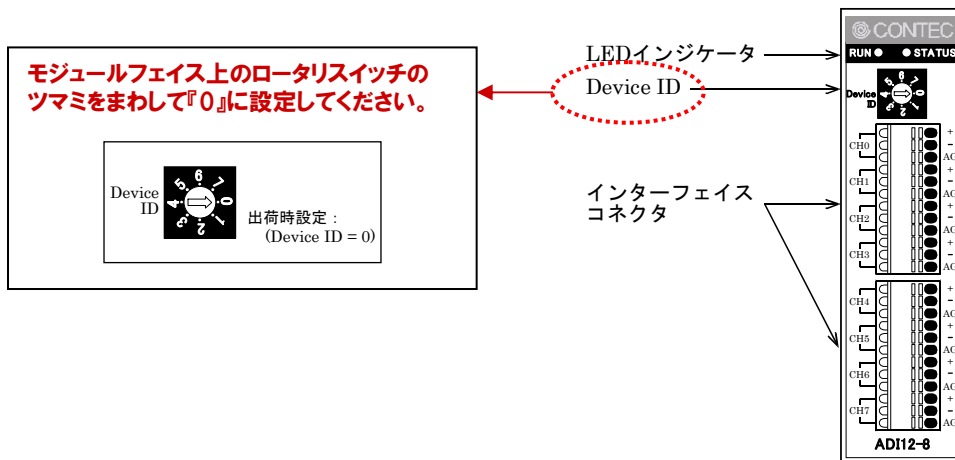
『Unit ID』は、同一『Group ID』内でI/Oコントローラユニットを区別するためのIDです。概念的には、『Group ID』の下位に属するものです。I/Oアシストサーバを使用しない場合、『Unit ID』によってI/Oコントローラユニットが区別され、『Unit ID』:0~7までの計8台が使用できます。本書では『Unit ID』は『0』に設定します。



② I/Oデバイスモジュール ADI12-8 (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、ADI12-8 (FIT) GYの『Device ID』は『0』とします。

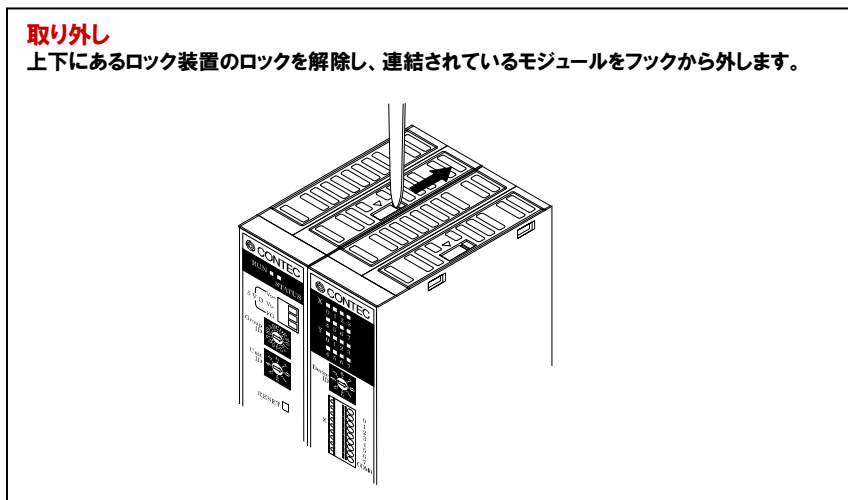
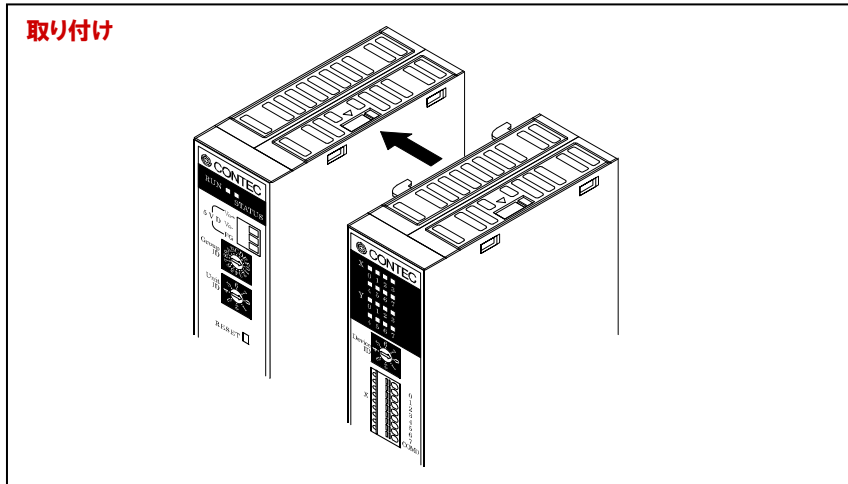


6-3-6.ハードウェアの準備 (I/Oデバイスモジュールの接続)

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。

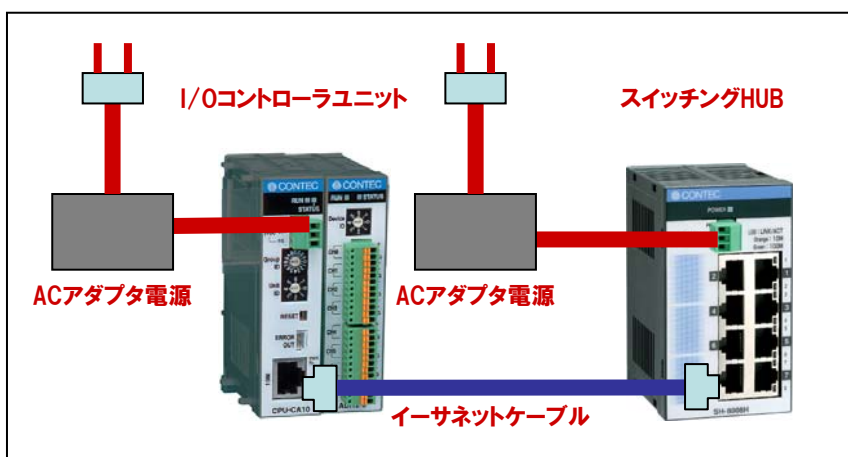
① I/OコントローラモジュールとI/Oデバイスモジュールとの接続

- (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witchングHUB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。



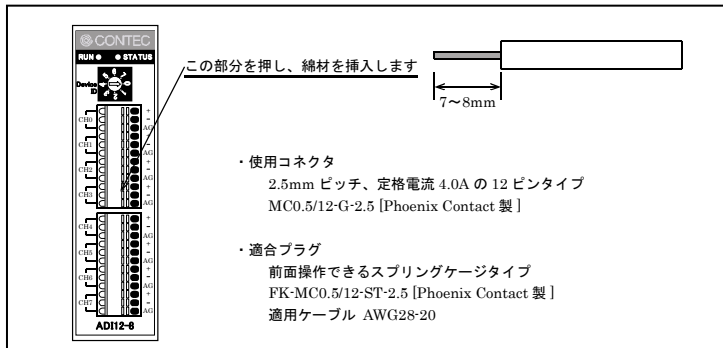
6-3-7.ハードウェアの準備 (I/Oデバイスモジュールと外部機器との接続)

I/Oデバイスモジュール (ADI12-8 (FIT) GY) と外部機器 (本書では、ファンクションジェネレータ) 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 (ADI12-8 (FIT) GY) と外部機器との接続

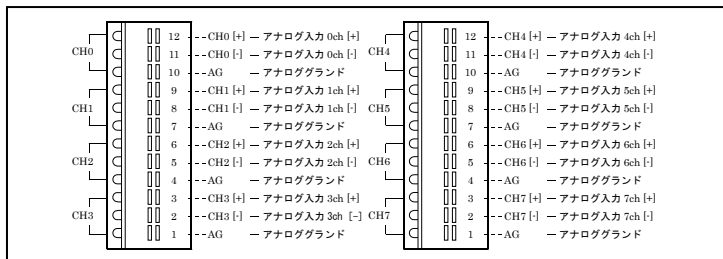
(a) 接続方法

ADI12-8 (FIT) GYと外部機器を接続する場合、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) インターフェースコネクタの信号配置

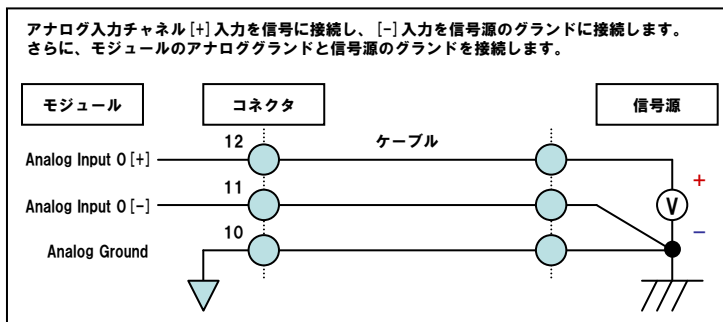
外部装置の接続は、モジュールのフェイスに装備された12ピン (1グループ) のコネクタで行います。



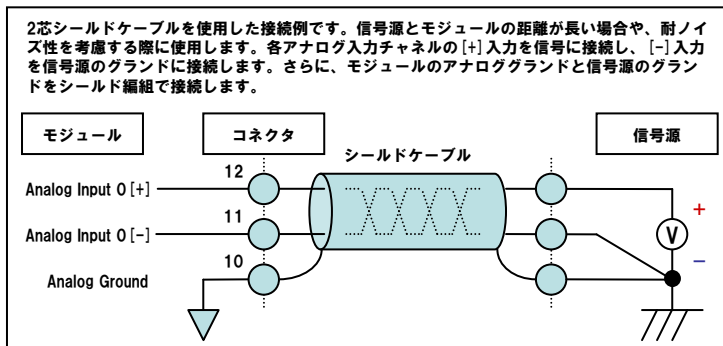
(c) 外部機器との接続例

本書のシステムでは、入力の0チャンネル (信号名:CH0) に外部機器を接続します。I/Oデバイスモジュールと外部機器との距離によって、フラットケーブル (近距離)、シールドケーブル (遠距離・耐ノイズ対策) を使い分けて下さい。

●フラットケーブルを使用する場合



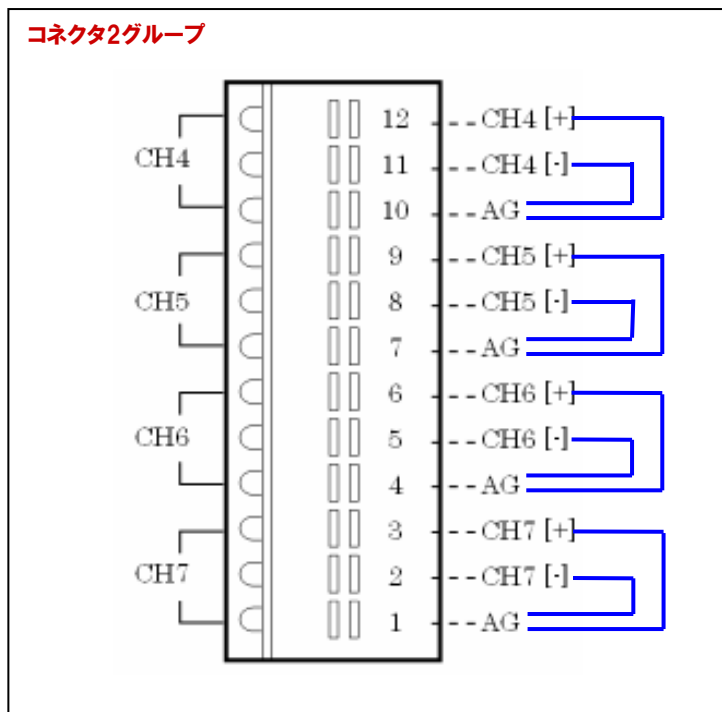
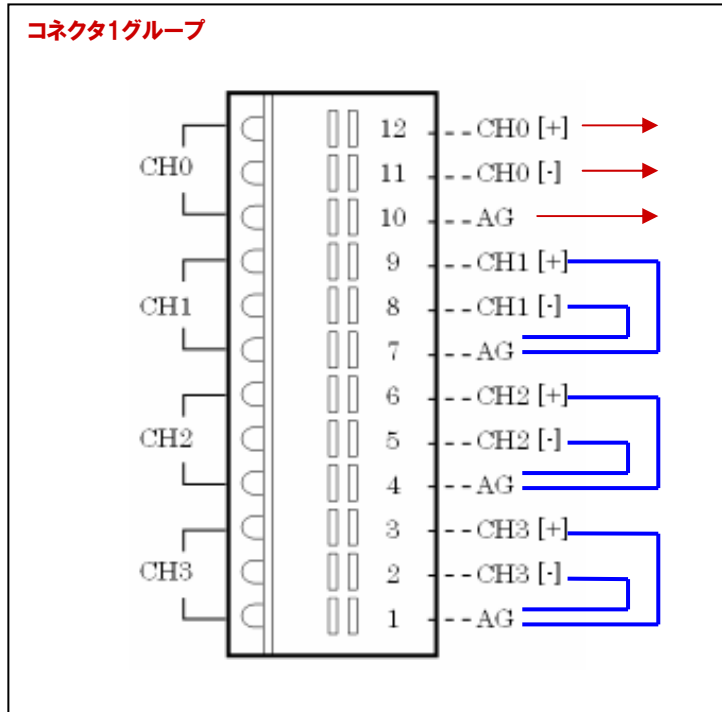
●シールドケーブルを使用する場合



(d) 未処理チャンネル端子の処理

今回使用するI/OデバイスモジュールADI12-8 (FIT) GYは、入力チャンネルを8チャンネル備えており、実習ではその中の『0チャンネル(信号名:CH0)』に外部機器を接続します。アナログ計測に於いて、信号端子に何も接続されていない(一般的に『浮いている』と言う)状態の場合、入力端子のインピーダンスが高いため、浮遊容量や他チャンネルからの影響(クロストーク)などにより、不定の値が計測される事が多くあります。信号源が接続されれば、正規の計測が行われますので問題は有りませんが、システム調整の段階に於いては接続ミスや断線との区別がつかず、現象にとまどうことも良くあります。

これを避けるためには、使用しないチャンネルの端子をすべてアナロググランドに接続しておく事をお勧めします。今回は、CH0だけの使用ですので、未使用のCH1からCH7の端子をすべてアナロググランドに接続します。



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

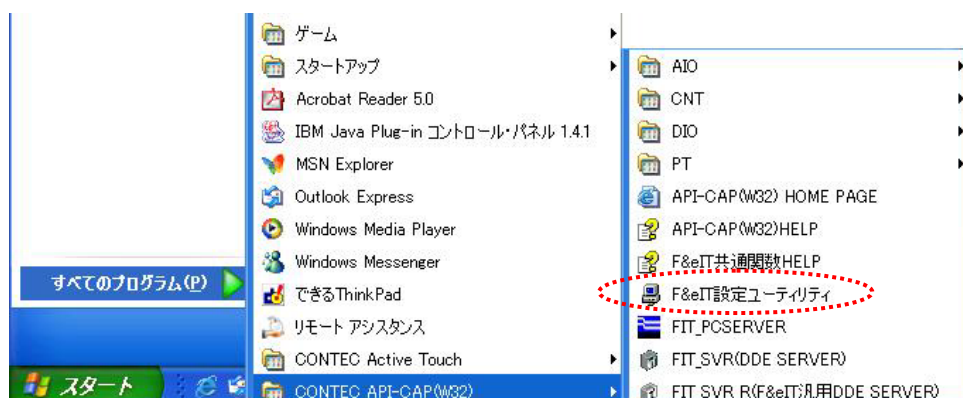
『00.80.4C』はベンダーコードと呼ばれ全機器共通になっており、IPアドレス生成では『10』として扱います。この機器の出荷時IPアドレスは、『00.80.4C』 = 『10』ですので、『10.01.02.03』に自動設定されています。なお、サブネットマスクは、『255.0.0.0』に設定されています。

- ① I/Oコントローラユニットの『Group ID』が『8』、『Unit ID』が『0』である事を確認します。
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アダプタとネットワークケーブルを接続します。

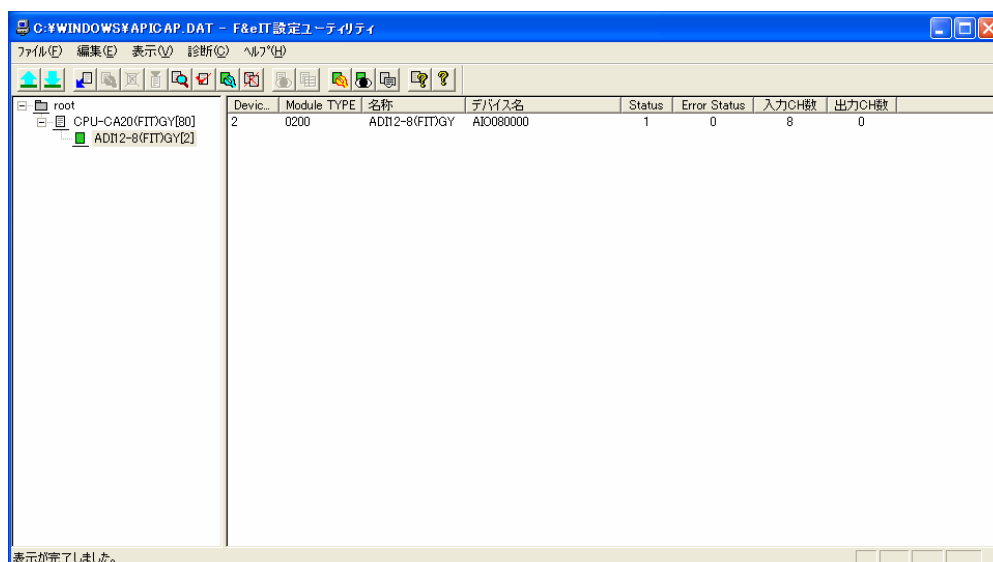
6-3-9.F&eIT設定ユーティリティによるセットアップ (ユーティリティの実行)

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。

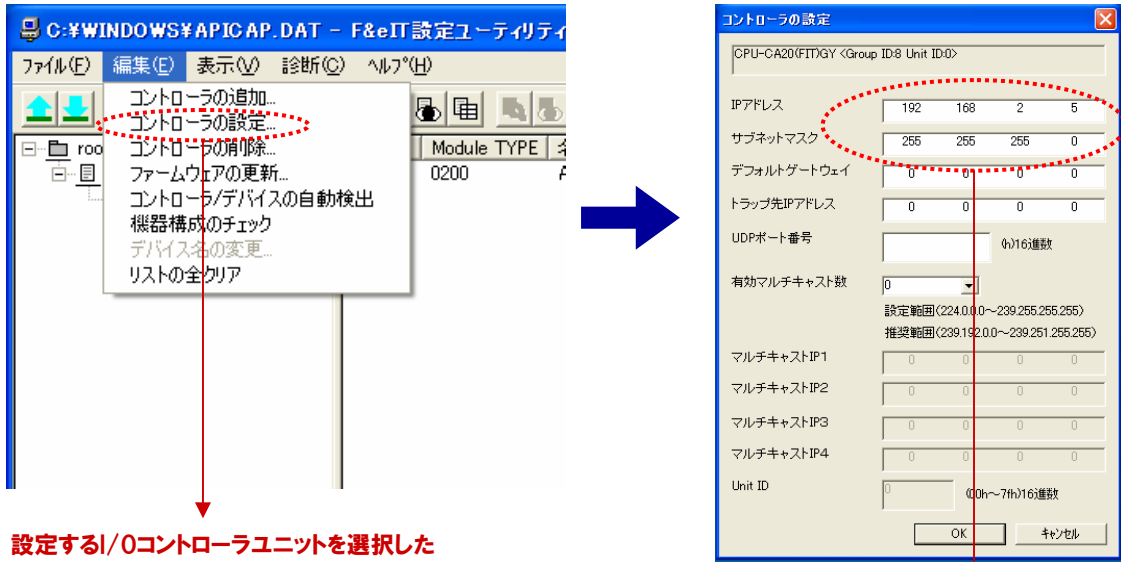


■ F&eIT設定ユーティリティのメイン画面



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。



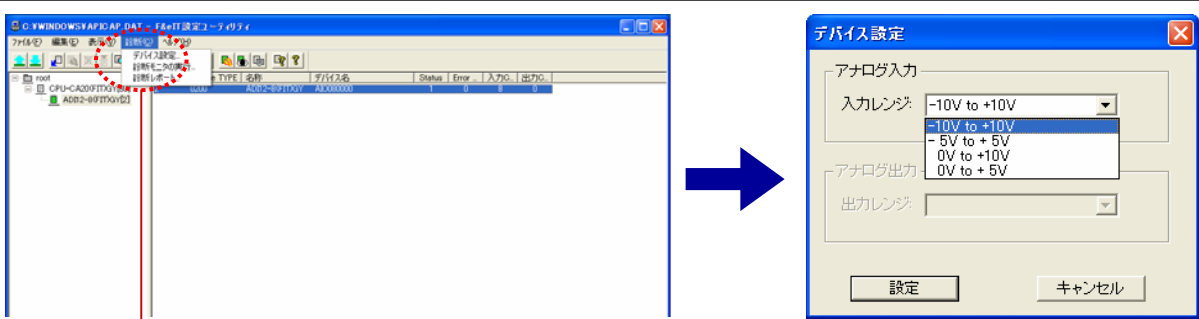
The screenshot shows the F&eIT utility interface. On the left, a tree view shows a controller selected. A context menu is open with 'コントローラの設定...' (Configure Controller...) highlighted. A red arrow points from this menu item to the 'コントローラの設定' dialog box on the right. The dialog box shows the configuration for 'GPU-CA20(FIT)GY <Group ID:8 Unit ID:0>'. The IP address is set to 192.168.2.5 and the subnet mask is 255.255.255.0. A red dashed circle highlights these two fields. Below the dialog box, a red arrow points to the text: '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'.

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)

アナログ入力デバイスにおける入力レンジ設定や、アナログ出力デバイスの出力レンジなど、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。



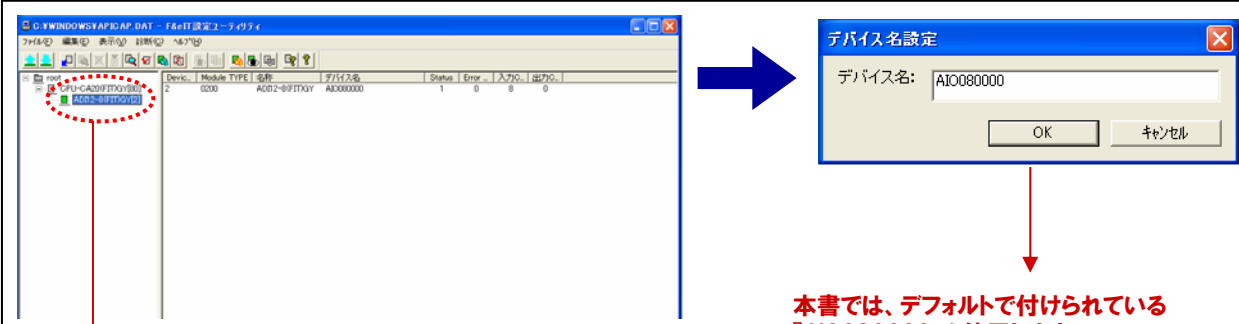
The screenshot shows the F&eIT utility interface. On the left, a tree view shows a device selected. A context menu is open with 'デバイスの設定...' (Configure Device...) highlighted. A red arrow points from this menu item to the 'デバイス設定' dialog box on the right. The dialog box shows the configuration for 'ADI12-8 (FIT) GY'. The input range is set to '-10V to +10V'. A red arrow points from the '-10V to +10V' option in the dropdown menu to the text: 'ADI12-8 (FIT) GYでは、入力レンジの設定を行うことができます。本書では『-10V to +10V』に設定します。'.

設定するI/Oデバイスモジュール(本書では、ADI12-8 (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。

ADI12-8 (FIT) GYでは、入力レンジの設定を行うことができます。
本書では『-10V to +10V』に設定します。

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。




設定するI/Oデバイスモジュール(本書では、ADI12-8 (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。

デバイス名設定
デバイス名: AIO080000
OK キャンセル

本書では、デフォルトで付けられている『AIO080000』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。

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。



設定ファイル再読み込み
設定ファイルの保存
アプリケーションの終了

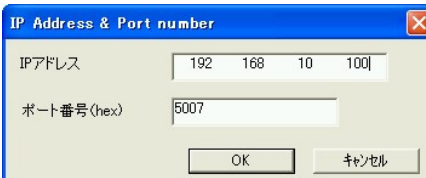
I/Oコントローラユニットを制御するパソコンで、使用するデバイスのデバイス名(デバイスの識別子)やIPアドレスを、設定ファイルとしてディスクに保存します。API関数ライブラリは、この設定ファイルを読みこんで動作します。

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こ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。



IP Address & Port number

IPアドレス: 192 168 10 100
ポート番号(hex): 5007
OK キャンセル

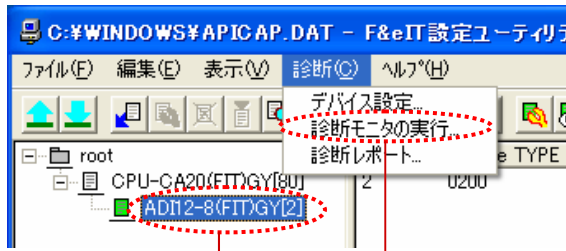
- ④『ファイル』メニューから『設定ファイルの保存』を実行してください。

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て、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。

■ アナログ診断モニタの実行



ADI12-8 (FIT) GYを
選択状態にします。

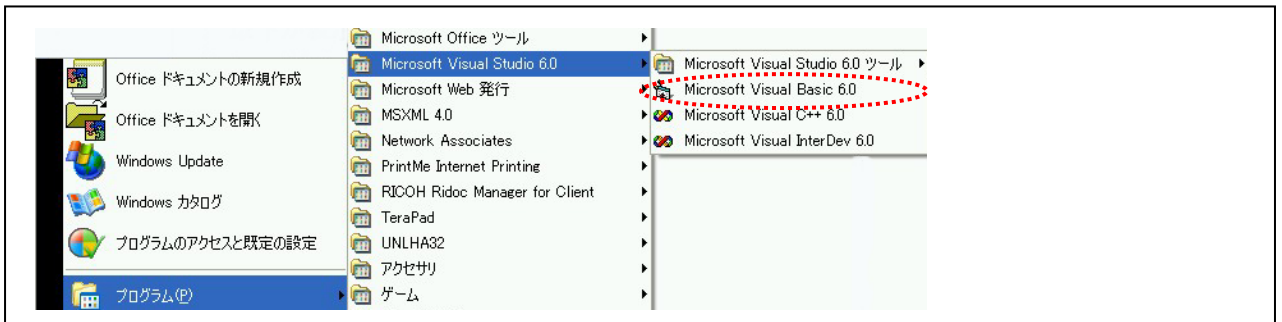
メニューの『診断』-
『診断モニタの実行』
を選択します。



6-3-15. Visual Basicの起動と画面の作成

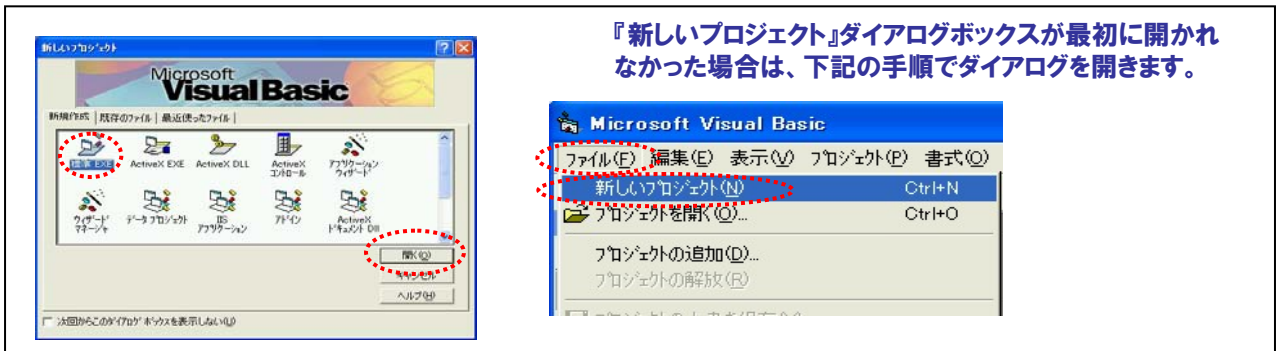
まず最初にVisual Basic6.0を起動しましょう。インストール時の設定が標準で、その後変更していなければ、次の手順で起動します。

『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。

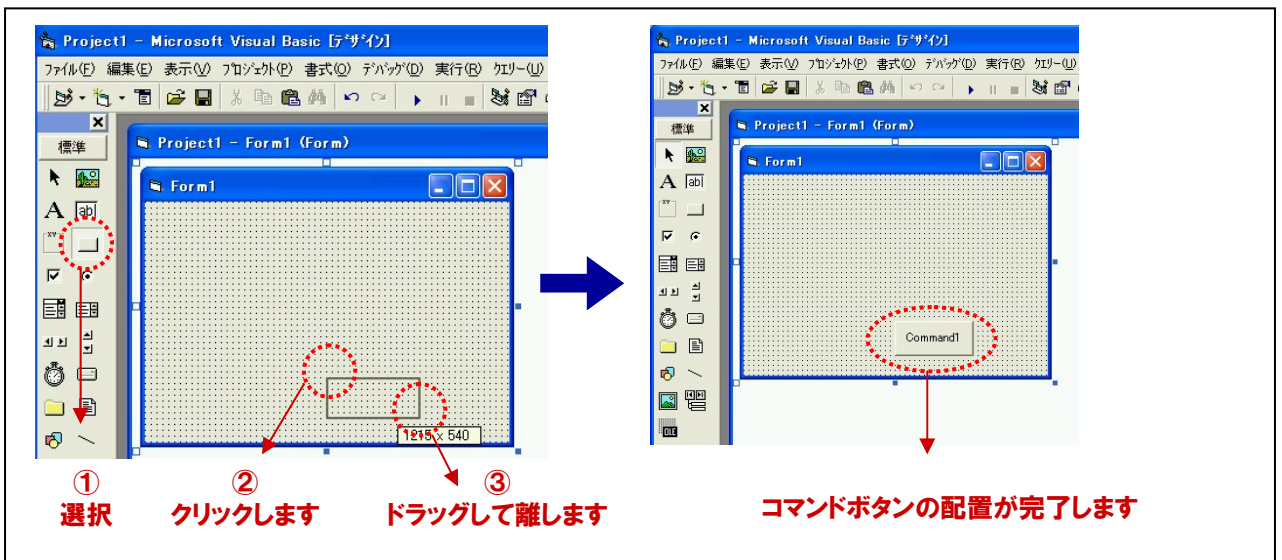
(※:『新しいプロジェクト』ダイアログボックスが表示されない場合は、メニューの『ファイル』 - 『新しいプロジェクト』を選択すると同様のダイアログボックスが表示されます)。



『新しいプロジェクト』ダイアログボックスが最初が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。

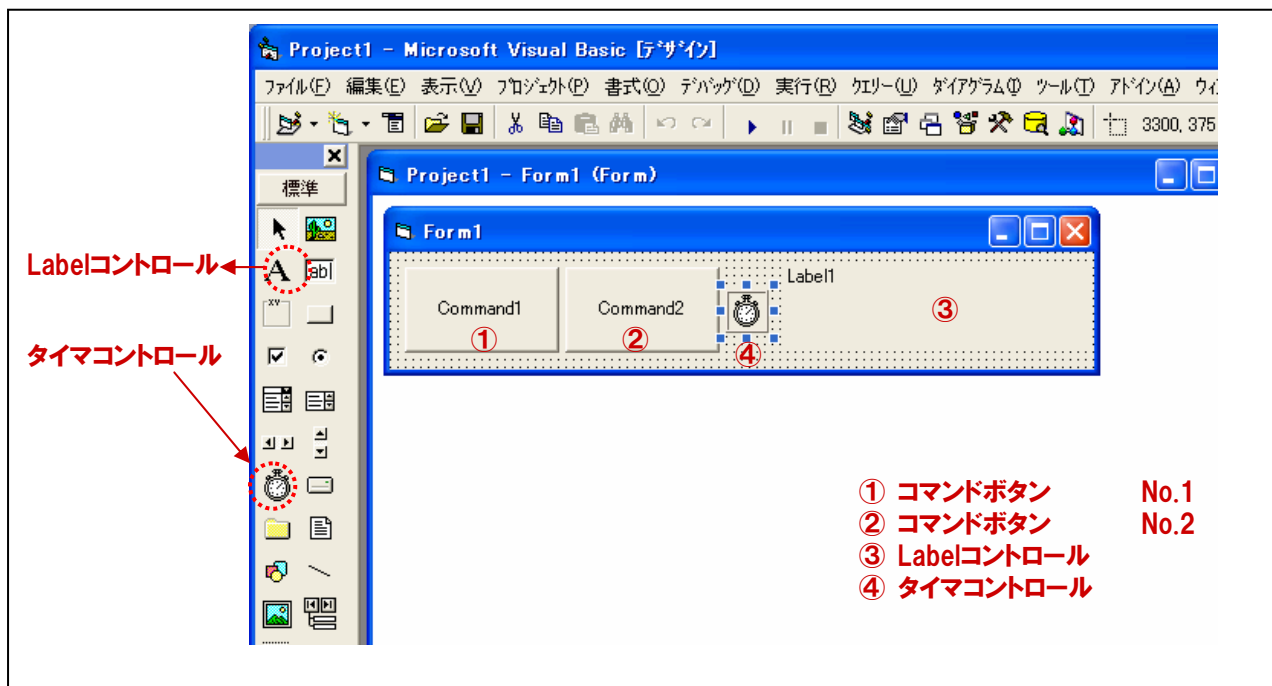
本プログラムにて使用する『コントロール』をフォーム上に配置します。まず最初に『コマンドボタン コントロール』をフォーム上に配置しましょう。

- 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ポインタ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

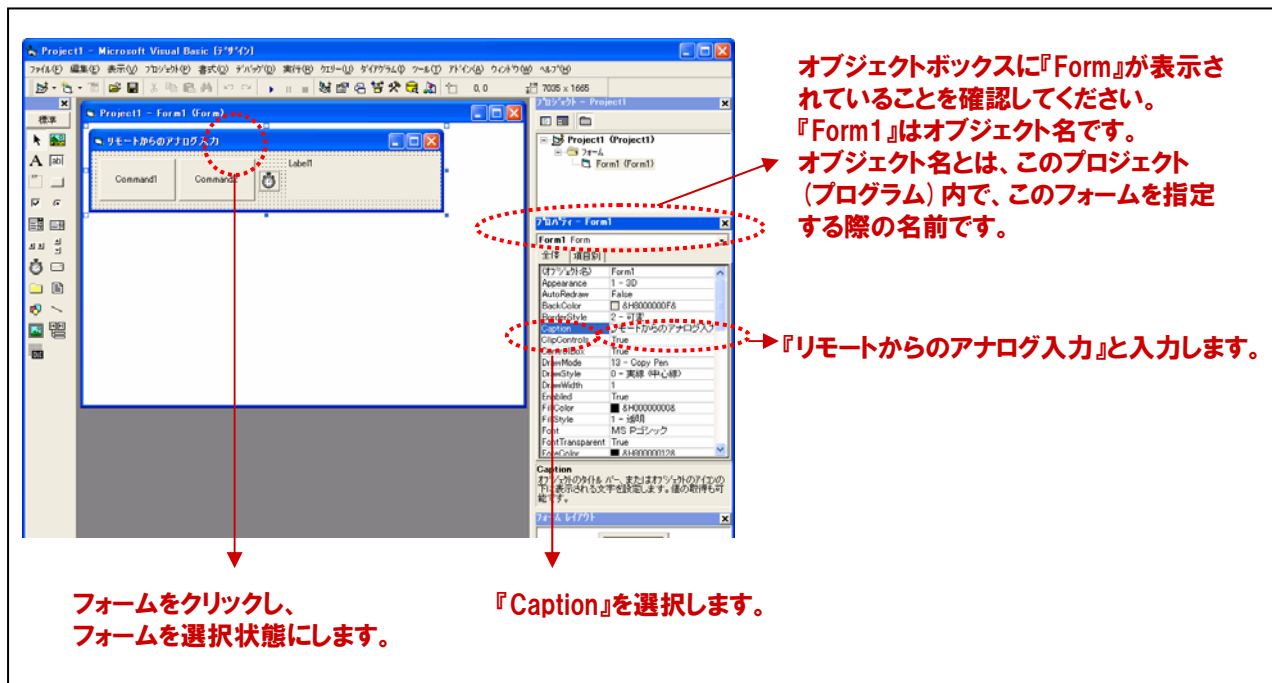
同様の手順で、2つ目の『コマンドボタン』と『Labelコントロール』、『タイマコントロール』をフォーム上に配置します。



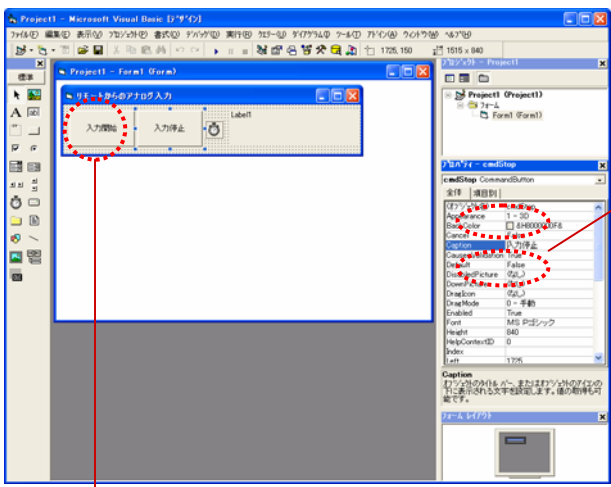
6-3-17.各オブジェクトのプロパティ設定

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『リモートからのアナログ入力』と入力します。



- 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、『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『入力開始』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『入力停止』と入力します。



コマンドボタン①のプロパティ設定

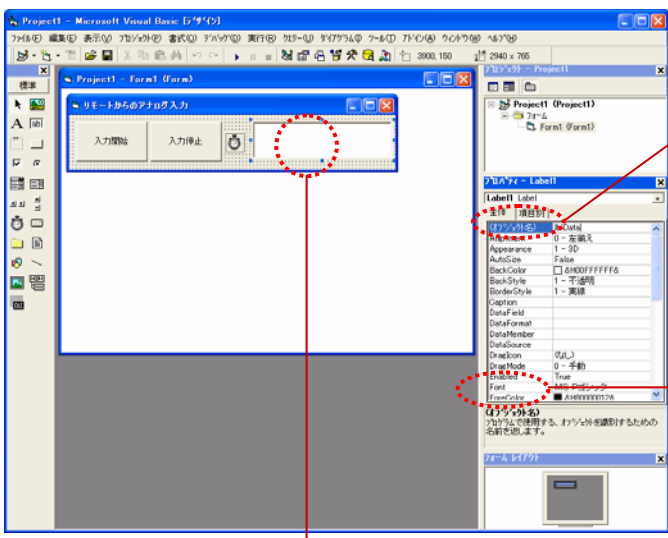
- オブジェクト名 : cmdStart
- Caption : 入力開始

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 : 入力停止

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』を変更します。Label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、『lblData』と入力します。本書では、その他に『Caption』=空白、『BackColor』=『白』、『BorderStyle』=『実線』を変更しています。また、『Font』プロパティを変更しています。『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。



Labelコントロールのプロパティ設定

- オブジェクト名 : lblData
- Caption : 空白
- BackColor : 白
- BorderStyle : 1-実線
- 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。

フォント

フォント名(F)	スタイル(S)	サイズ(Sz)
MS Pゴシック	標準	9
MS Pゴシック	標準	9
MS Pゴシック	斜体	10
MS Reference Sans Serif	太字	11
MS Reference Specialty	太字 斜体	12
MS Sans Serif		14
MS Serif		16
MS UI Gothic		18

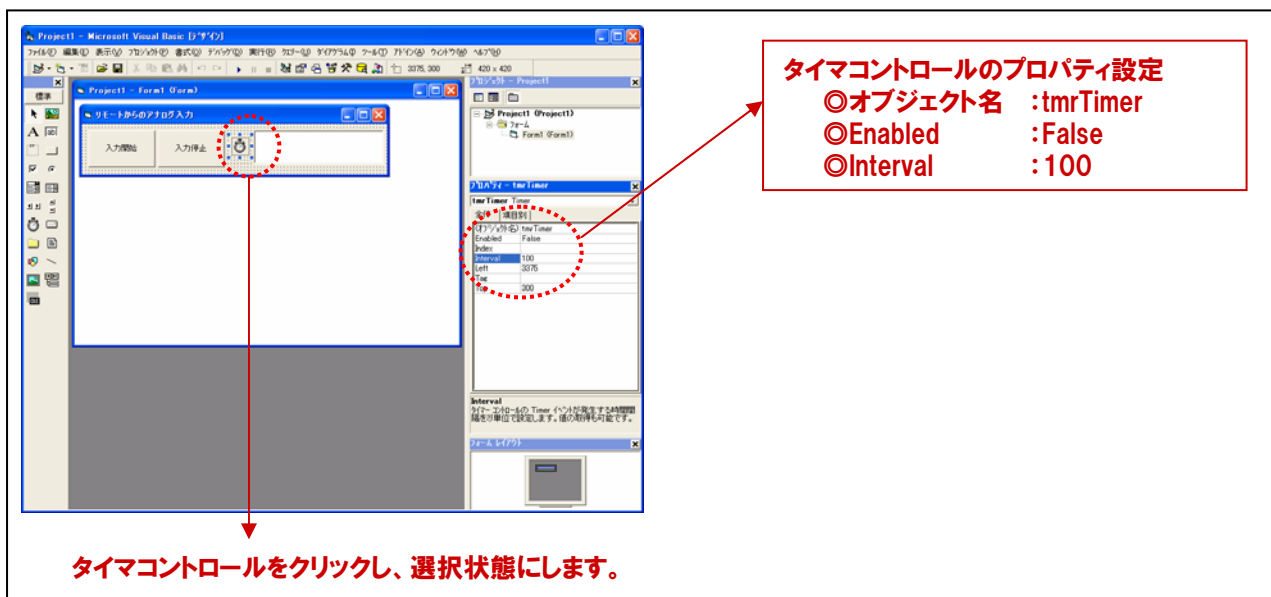
文字種(C)
 期的消し線(L)
 下線(U)

サンプル: AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz

文字セット(O): 日本語

Labelコントロールをクリックし、選択状態にします。

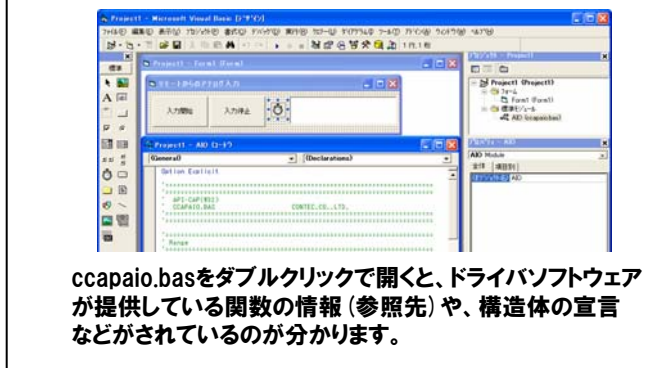
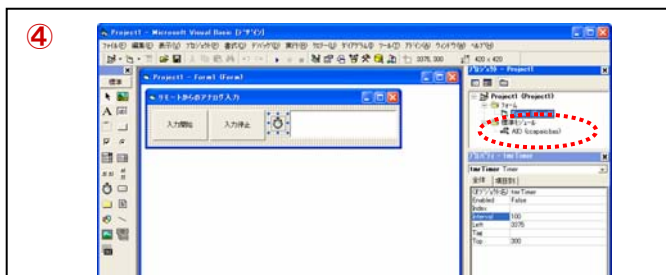
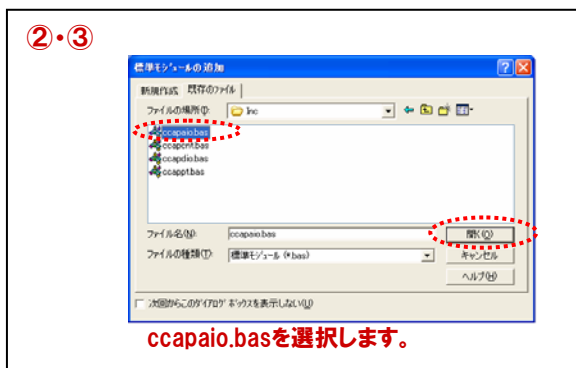
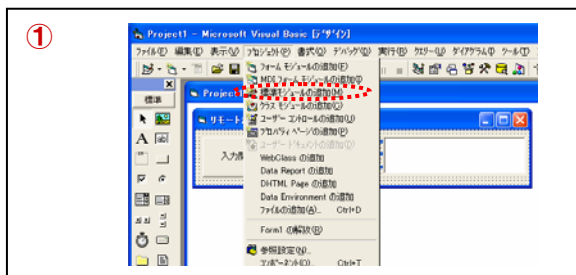
- ④ タイマコントロールの『オブジェクト名』と『Enabled』、『Interval』プロパティを変更します。タイマコントロールを選択、プロパティウィンドウのプロパティリスト『(オブジェクト名)』を選択して、『tmrTimer』と入力します。続いて、プロパティリストの中から、『Enabled』プロパティを選択して、コンボボックスから『False』を選択します。最後に『Interval』に『100』と入力します。



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。

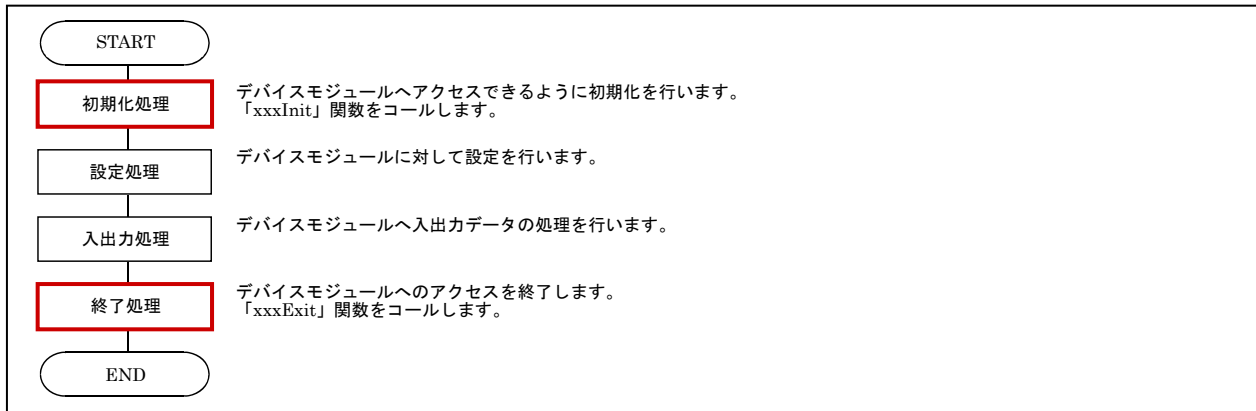
- ① Visual Basicのメニュー『プロジェクト』 - 『標準モジュールの追加』を選択します。
- ② ダイアログが表示されますので、『既存のファイル』のタブをクリックします。
- ③ 標準設定でインストールした場合には下記の場所に標準モジュールファイルがあります。本書では、アナログ入出力の機能を使用しますので『ccapaio.bas』を選択して『開く』ボタンをクリックします。
C:¥Program Files¥CONTEC¥API-CAP (W32) ¥Samples¥Inc¥ccapaio.bas
- ④ プロジェクトエクスプローラ上に標準モジュールファイルが追加されていることを確認してください。



6-3-19.API-CAP (W32) の処理体系

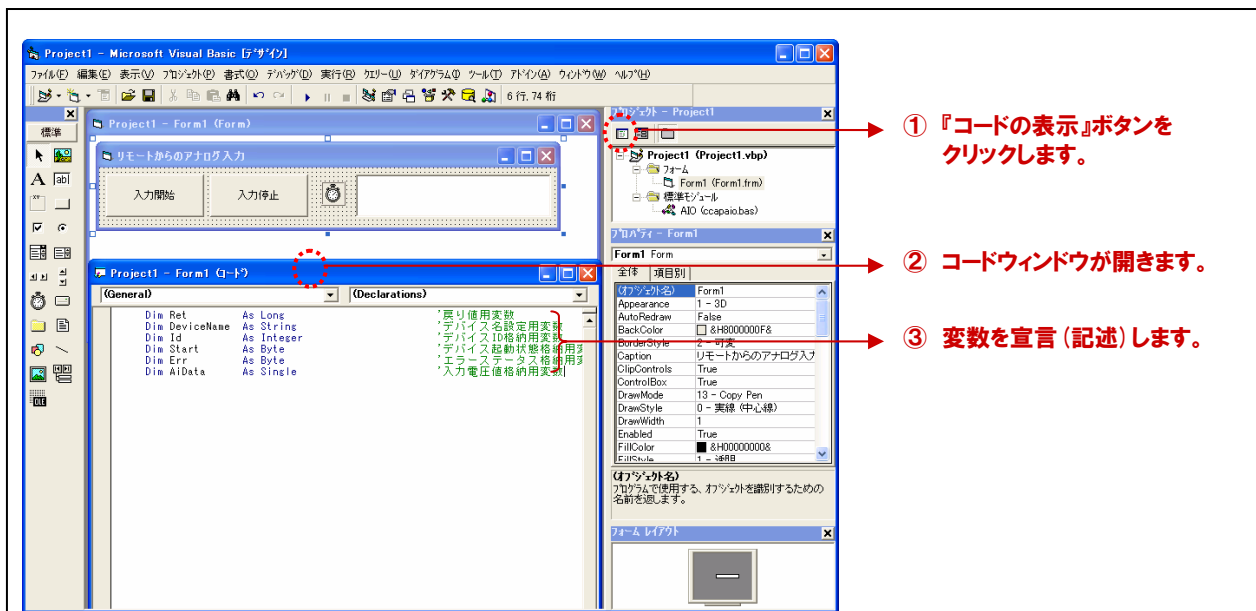
API-CAP (W32) は、初期化処理ではじまり、終了処理で終了する決まりがあります。初期化処理および、終了処理の専用関数を用意していますので、それぞれのタイミングでその関数を実行します。

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。



Dim Ret	As Long	'戻り値用変数
Dim DeviceName	As String	'デバイス名設定用変数
Dim Id	As Integer	'デバイスID格納用変数
Dim Start	As Byte	'デバイス起動状態格納用変数
Dim Err	As Byte	'エラーステータス格納用変数
Dim AiData	As Single	'入力電圧値格納用変数

6-3-21.初期化処理・入力レンジ設定・デバイス起動状態の取得と起動処理の記述

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。

また、使用するアナログ入出力デバイスモジュールの入力レンジ設定および、現在のアナログ入出力の起動状態を確認し、デバイスモジュールが停止状態であれば、デバイスモジュールを起動する処理を行います。

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の間に各処理を記述します。

```
DeviceName = "AI0080000"
Ret = AioInit (DeviceName, Id)
```

デバイス名を変数に格納
初期化処理

```
Ret = AioSetAiRangeAll (Id, PM10)
```

入力レンジ設定 (PM10: ±10V)

```
Ret = AioGetRemoteStatus (Id, Start, Err)
```

デバイスの状態を取得

```
If Start = &H0 Then
    Ret = AioStartIO (Id, &H1)
End If
```

デバイスが停止状態の場合 (停止: 0)
デバイスを起動 (起動: 1)

初期化処理関数『AioInit』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。AioInitが正常終了した後、AioExit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 Id As Integer
Dim DeviceName As String
Ret = AioInit (DeviceName, Id)
```

■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。

Id : 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、割愛しています。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。

全チャンネルのアナログ入力レンジ設定関数『AioSetAiRangeAll』リファレンス

■機能 全チャンネルに対してアナログ入力レンジの設定を行います。

■書式 Visual Basic6.0の場合

```
Dim Id As Integer
Dim AiRange As Integer
Ret = AioSetAiRangeAll (Id, AiRange)
```

■引数 Id : Aiolnit 関数で取得したIDを指定します。

AiRange : アナログ入力レンジを以下の範囲からマクロもしくは数値で指定します。設定できる値はデバイスにより異なります。

レンジ	マクロ	値
±10V	PM10	0
±5V	PM5	1
0~10V	P10	50
0~5V	P5	51
0~20mA	P20MA	100

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、割愛しています。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 本書で使用しているADI12-8 (FIT) GYは、±10V、±5V、0~10V、0~5Vの設定が可能です。チャンネル単位での設定はできません。デバイスモジュール単位での設定になります。

機器状態の取得関数『AioGetRemoteStatus』リファレンス

■機能 デバイスのエラーステータスを返します。

■書式 Visual Basic6.0の場合

```
Dim Id As Integer
Dim Start As Byte
Dim Status As Byte
Ret = AioGetRemoteStatus (Id,Start,Status)
```

■引数 Id : Aiolnit 関数で取得したIDを指定します。

Start : 起動レジスタを返す変数を指定します。『0:停止』、『1:起動』です。

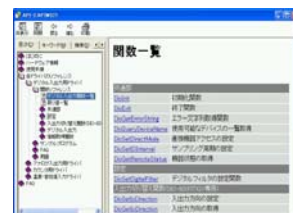
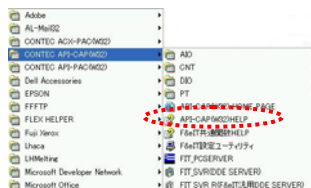
Status : ステータスを返す変数を指定します。ステータスは0以外であれば、エラーです。

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、割愛しています。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。



サンプリングの開始／停止処理関数 『AioStartIO』 リファレンス

■機能 アナログ値の取得を開始／停止します。

■書式 Visual Basic6.0の場合

```
Dim Id As Integer
Dim Start As Byte
Ret = AioStartIO (Id,Start)
```

■引数 Id : Aiolnit 関数で取得したIDを指定します。

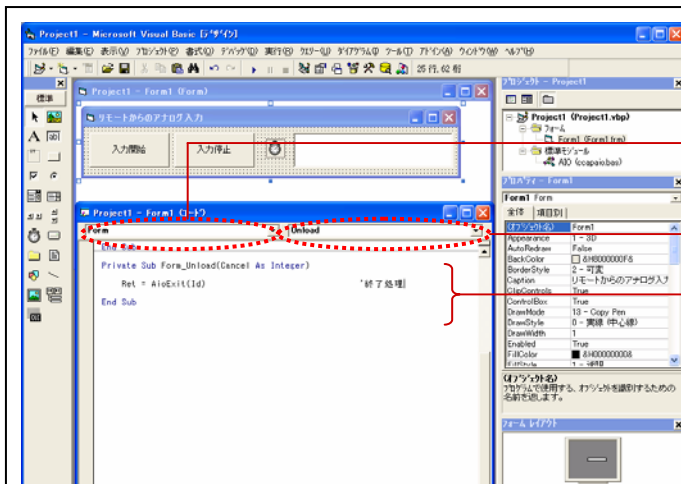
Start : 『開始の場合:1』、『停止の場合:0』を指定します。

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、割愛しています。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。

6-3-22.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 = AioExit (Id)
```

‘終了処理実行

終了処理関数 『AioExit』 リファレンス

■機能 ドライバの終了処理を行います。アプリケーションの終了時に実行します。ドライバが使用していたメモリ、スレッドをすべて開放します。実行後は、Aiolnit関数を実行するまでデバイスにアクセスできません。

■書式 Visual Basic6.0の場合

```
Dim Id As Integer
Ret = AioExit (Id)
```

■引数 Id : Aiolnit (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、割愛しています。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。

100msec (0.1秒) 毎にアナログ入出力処理を行うために、【4-2.】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【4-2.】と同様に『入力開始ボタン』で開始、『入力停止ボタン』で停止の処理を行います。

6-3-23.タイマ開始処理の記述

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。『入力開始ボタン』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『入力開始ボタン』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 『入力開始ボタン』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
```

『タイマ起動』

6-3-24.タイマ停止処理の記述

『入力開始ボタン』で有効にしたタイマコントロールを停止させるための『入力停止ボタン (cmdStop)』の処理を記述していきます。この『入力停止ボタン』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『入力停止ボタン』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 『入力停止ボタン』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。

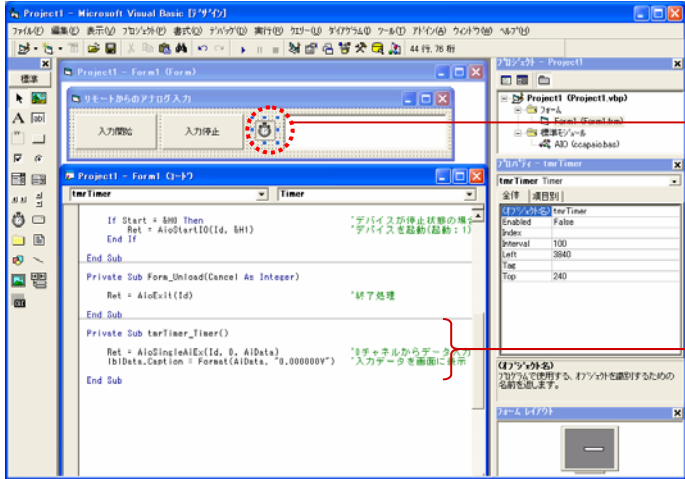
Private Sub cmdStop_Click () からEnd Subまでの間に、下記のコードを記述します

```
tmrTimer.Enabled = False
```

『タイマ停止』

6-3-25.アナログ入力および表示処理の記述

ADI12-8 (FIT) GYの『入力0ch』からアナログデータの入力を行う処理を追加します。API-CAP (W32) では、電圧/電流値、バイナリ値でアナログデータを入力する関数が提供されています。今回は、電圧/電流値でデータを入力する関数を使用して、その結果を画面にリアルタイムに表示します。



① 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。

② Private Sub tmrTimer_Timer () から End Subの間に入力処理関数と、画面表示処理を記述します。

```
Ret = AioSingleAiEx (Id, 0, AiData)
lblData.Caption = Format (AiData, "0.000000V")
```

'0チャンネルからデータ入力 (電圧値)
'入力データを画面に表示

指定チャンネルの電圧/電流値入力処理関数 『AioSingleAiEx』 リファレンス

■機能 指定チャンネルを1回A/D変換し、変換データを電圧または電流で返します。

■書式 Visual Basic6.0の場合

```
Dim Id As Integer
Dim AiChannel As Integer
Dim AiData As Single
Ret = AioSingleAiEx (Id, AiChannel, AiData)
```

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

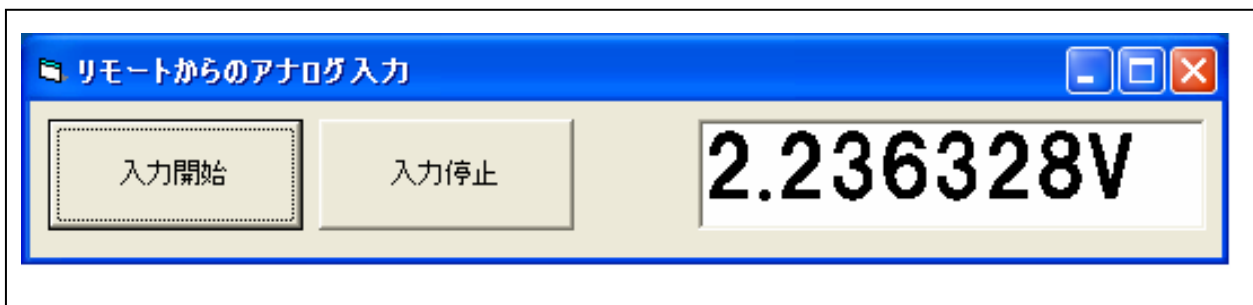
AiChannel : 変換チャンネルを指定します。

AiData : 変換データを格納する変数を指定します。変換データは電圧または電流値で格納されます。

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、割愛しています。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。

6-3-26.プログラムの実行イメージ



6-4.アナログ入出力カード (ADA16-8/2 (CB) L) とのプログラム比較

今回作成したリモートI/Oでのアナログ入出力と、第4章で作成したADA16-8/2 (CB) L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。

①F&EIT機器とAPI-CAP (W32) を使用したアナログ入力プログラミング例 (今回のプログラムリスト)

```
Dim Ret As Long '戻り値用変数
Dim DeviceName As String 'デバイス名設定用変数
Dim Id As Integer 'デバイスID格納用変数
Dim Start As Byte 'デバイス起動状態格納用変数
Dim Err As Byte 'エラーステータス格納用変数
Dim AiData As Single '入力電圧値格納用変数

Private Sub Form_Load ()

    DeviceName = "AI080000" 'デバイス名を変数に格納
    Ret = AioInit (DeviceName, Id) '初期化処理

    Ret = AioSetAiRangeAll (Id, PM10) '入力レンジ設定 (PM10:±10V)

    Ret = AioGetRemoteStatus (Id, Start, Err) 'デバイスの状態を取得

    If Start = &H0 Then 'デバイスが停止状態の場合 (停止:0)
        Ret = AioStartIO (Id, &H1) 'デバイスを起動 (起動:1)
    End If

End Sub

Private Sub Form_Unload (Cancel As Integer)

    Ret = AioExit (Id) '終了処理

End Sub

Private Sub cmdStart_Click ()

    tmrTimer.Enabled = True 'タイマ起動

End Sub

Private Sub cmdStop_Click ()

    tmrTimer.Enabled = False 'タイマ停止

End Sub

Private Sub tmrTimer_Timer ()

    Ret = AioSingleAiEx (Id, 0, AiData) '0チャンネルからデータ入力 (電圧値)
    lblData.Caption = Format (AiData, "0.000000V") '入力データを画面に表示

End Sub
```

②ADA16-8/2 (CB) LとAPI-PAC (W32) を使用したアナログ入力プログラミング例 (第4章)

Dim Ret	As Long	'戻り値用変数
Dim DeviceName	As String	'デバイス名設定用変数
Dim Id	As Integer	'デバイスID格納用変数
Dim AiData	As Single	'入力電圧値格納用変数

Private Sub Form_Load ()

DeviceName = "AI0000"	'デバイス名を変数に格納
Ret = AioInit (DeviceName, Id)	'初期化処理 (デバイスハンドル取得)

Ret = AioSetAiRangeAll (Id, PM10)	'入力レンジ設定 (入力レンジ±10V)
-----------------------------------	----------------------

End Sub

Private Sub Form_Unload (Cancel As Integer)

Ret = AioExit (Id)	'終了処理 (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 ()

Ret = AioSingleAiEx (Id, 0, AiData)	'入力0chのデータを変数AiDataに格納
lblData.Caption = Format (AiData, "0.000000V")	'変数AiDataを表示

End Sub

6-5.API-CAP (W32) アナログ入出力用ドライバ 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。本書にて使用したアナログ入出力用ドライバには、弊社製アナログ入出力リモート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

AioInit	初期化関数
AioExit	終了関数
AioGetErrorString	エラー文字列取得関数
AioQueryDeviceName	使用可能なデバイスの一覧取得
AioGetDeviceType	デバイスの種類を取得
AioSetDirectMode	直接機器アクセスの設定
AioSetIOInterval	サンプリング周期の設定
AioStartIO	サンプリングの開始／停止
AioGetRemoteStatus	機器状態の取得

②設定

AioSetAiRangeAll	全チャンネルのアナログ入力レンジを設定
AioGetAiRange	指定チャンネルのアナログ入力レンジを取得
AioSetAoRangeAll	全チャンネルのアナログ出力レンジを設定
AioGetAoRange	指定チャンネルのアナログ出力レンジを取得

③アナログ入力

AioSingleAi	指定チャンネルを1回バイナリ値で入力
AioSingleAiEx	指定チャンネルを1回電圧または電流値で入力
AioMultiAi	複数チャンネルを1回バイナリ値で入力
AioMultiAiEx	複数チャンネルを1回電圧または電流値で入力

④アナログ出力

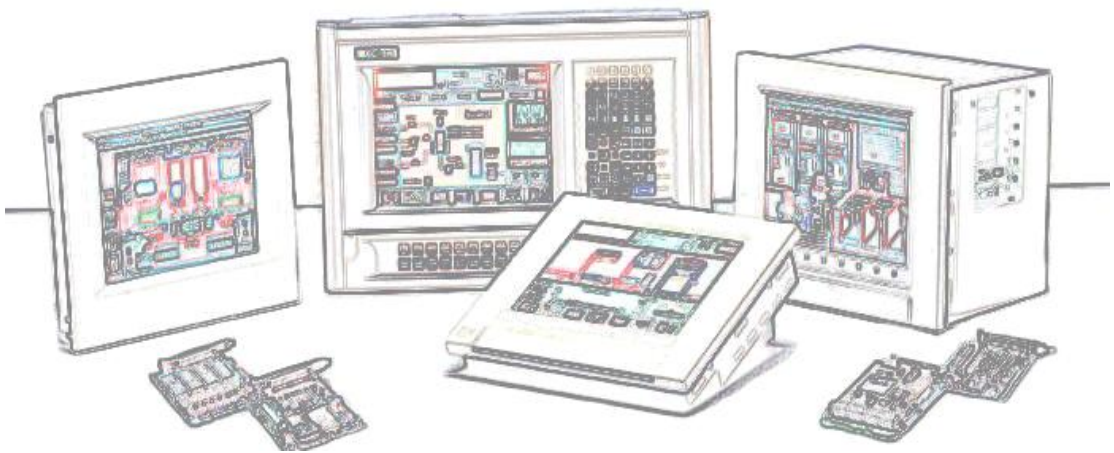
AioSingleAo	指定チャンネルを1回出力 (バイナリ値指定)
AioSingleAoEx	指定チャンネルを1回出力 (電圧／電流値指定)
AioSingleAoEcho	指定チャンネルの出力値を取得
AioMultiAo	複数チャンネルを1回出力 (バイナリ値指定)
AioMultiAoEx	複数チャンネルを1回出力 (電圧／電流値指定)
AioMultiAoEcho	複数チャンネルの出力値を取得

⑤情報取得関数

AioGetAiResolution	アナログ入力の分解能を取得
AioGetAiMaxChannels	アナログ入力の最大使用可能チャンネル数の取得
AioGetAoResolution	アナログ出力の分解能を取得
AioGetAoMaxChannels	アナログ出力の最大使用可能チャンネル数の取得

付録

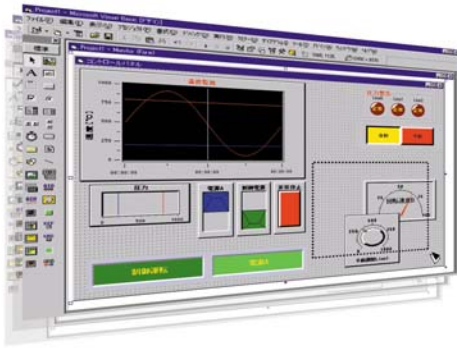
各種開発ツール紹介



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

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。
GPIO通信	弊社製GPIO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"> Microsoft Windows XP Professional Microsoft Windows XP Home Edition Microsoft Windows 2000 Professional Microsoft Windows 2003 Server Microsoft Windows NT Ver. 4.0 (SP3 以上) + Internet Explorer 4.01以上 Microsoft Windows Me Microsoft Windows 98およびSecond Edition Microsoft Windows 95 (SP1 以上) + Internet Explorer 4.01以上 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"> Microsoft Visual Basic Ver.6.0, 5.0 Microsoft Visual C++ Ver.6.0, 5.0 Microsoft Visual Basic 2005, .NET 2003, .NET 2002 Microsoft Visual C++ 2005, .NET 2003, .NET 2002 Microsoft Visual C# 2005, .NET 2003, .NET 2002 Microsoft Excel 2003 (VBA 6.4)、2002 (VBA 6.3)、2000 (VBA 6.0)、97 (VBA 5.0) Borland Delphi Ver.7, Ver.5, Ver.4 National Instruments LabVIEW 8, 7.1, 7.0, 6.1, 6i 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"> IBM PC/AT 互換機、DOS/V 機
その他
<ul style="list-style-type: none"> Pentium100MHz以上のCPUを推奨 プログラミング言語(コンテナ)が正常に動作する環境

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。

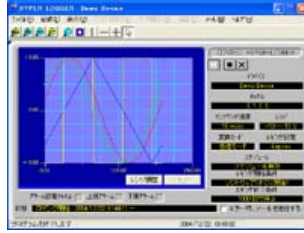
各種設定はプロパティページにより、プログラムレスで行えます。

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！

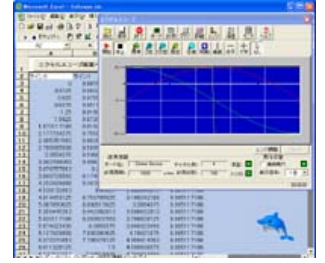
充実の実例集 (実用アプリケーション・プログラム)

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。

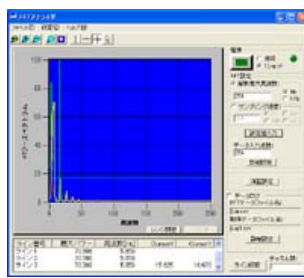
- **ハイパーロガー**
アナログ信号の高速連続サンプリングとファイリングができます。



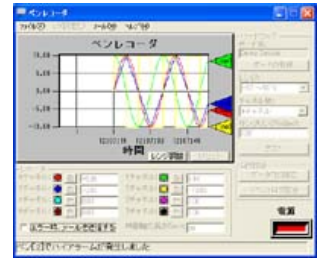
- **エクセルスコープ**
アナログ信号をサンプリングし、フォームウィンドウにグラフ表示するとともにワークシートに書き出します。
※Microsoft Excel 2003,2002,2000または97対応



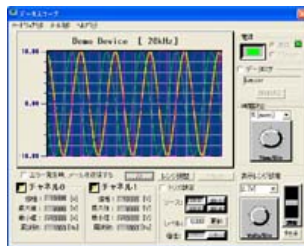
- **FFTアナライザ**
アナログ信号をサンプリングし、FFT解析を行うことができます。



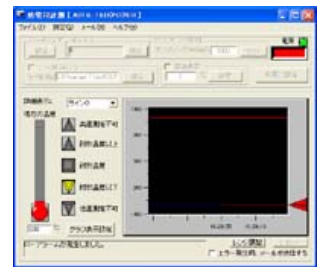
- **ペンレコーダ**
複数のアナログ信号をサンプリングし、信号をペンレコーダのようにグラフ描画していきます。



- **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。



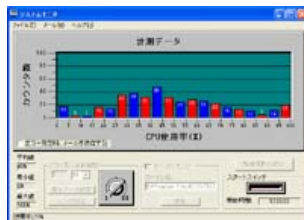
- **デジタルモニタ**
デジタル入出力とスイッチ、ランプを組み合わせた表示例です。



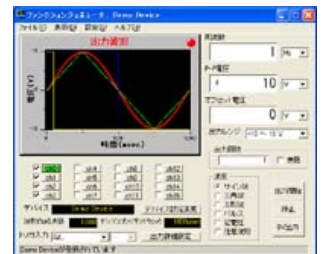
- **カウンタモニタ**
カウンタ入力および周波数を計測し、表示します。



- **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。



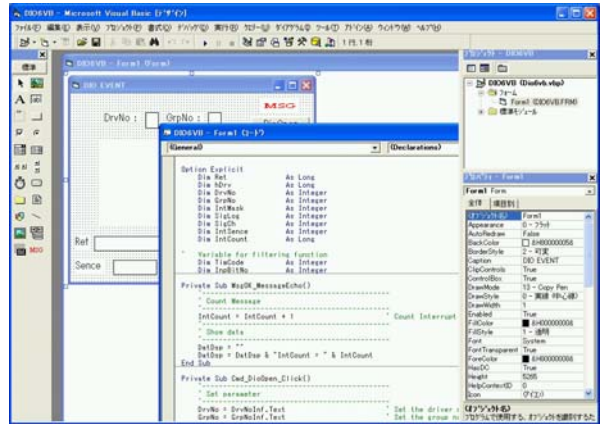
ACX-PAC (W32) の最新情報は、下記URLへアクセスしてください！
<http://www.contec.co.jp/acxpac/>

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
GPIO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)	対応開発環境
<ul style="list-style-type: none">Microsoft Windows XP ProfessionalMicrosoft Windows XP Home EditionMicrosoft Windows 2000 ProfessionalMicrosoft Windows NT 4.0Microsoft Windows NT 3.51Microsoft Windows MeMicrosoft Windows 98およびSecond EditionMicrosoft Windows 詳しくは、弊社ホームページをご確認ください。	<ul style="list-style-type: none">Visual C++ Ver.6.0, 5.0, 4.x, 2.0Borland C++ Ver.5.0, 4.5xVisual Basic Ver.6.0, 5.0, 4.0 (32ビットのみ)Visual C++ .NET 2003, 2002Visual C# .NET 2003, 2002Visual Basic .NET 2003, 2002Borland C++ Builder 6.0, 5.0Borland 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">● 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。● 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。● 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。
アナログ入出力ドライバ API-AIO (LNX)	<ul style="list-style-type: none">● 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。● アナログ入出力の基本的な機能を提供しています。● 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。● 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。● 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。
GPIB通信ドライバ API-GPIB (LNX)	<ul style="list-style-type: none">● 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。● IEEE-488規格に準拠しています。● 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こなします。
カウンタ入力ドライバ API-CNT (LNX)	<ul style="list-style-type: none">● 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。● モード設定、カウンタ値取得、カウンタ一致割り込み、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。
汎用入出力ドライバ IO-LIB (LNX) ※1	<ul style="list-style-type: none">● 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。● 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。● 割り込みイベント処理を行うことができます。● コンソールおよびX-Window (kylx) のサンプルプログラムを付属しています。● HTML形式の関数リファレンスを付属しています。● 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。

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">● gcc● kylx2 ※2	<ul style="list-style-type: none">● 2.4.21 / RedHat Linux Professional Workstation● 2.4.20 / RedHat Linux 9● 2.4.18 / RedHat Linux 8.0● 2.4.18 / RedHat Linux 7.3● 2.4.7 / RedHat Linux 7.2● 2.4.2 / RedHat Linux 7.1● 2.2.16 / RedHat Linux 7.0● 2.2.14 / RedHat Linux 6.2● 2.4.18 / TurboLinux 8● 2.4.5 / TurboLinux 7.0● 2.2.13 / TurboLinux 6.0

※1 : API-DIO (LNX) 、 IO-LIB (LNX) は、Kernel2.6.xx、RedHat Enterprise Linux4、Turbo Linux10Serverに対応しています。

※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

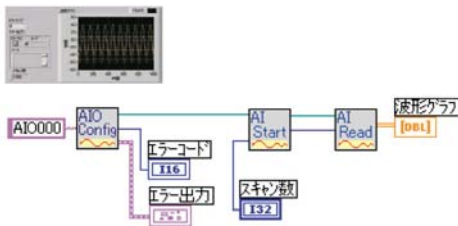
LabVIEW 対応データ集録用 VI ライブラリ VI-DAQ を使用する

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

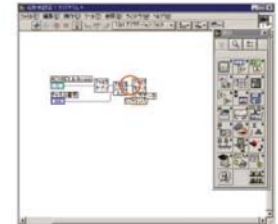
National Instruments 社の LabVIEW で弊社製 GPIB 通信ボード/カードを使用するためのドライバソフトウェアです。本ソフトウェアをインストールすることにより、弊社製 GPIB 通信ボード/カードを使用した LabVIEW でのプログラム開発や、完成した GPIB 通信プログラムを弊社製 GPIB 通信ボード/カードを使用して動作させることが可能です。また、National Instruments 社の API 関数スタイルで作成されているため、Microsoft Visual Basic をはじめとする他のプログラム言語で使用することもできます。



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 NT Workstation 4.0 + SP3 以降、Windows Me/98SE/98/95

■対応言語

National Instruments LabVIEW 7.1 / 7.0 / 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。
*2: National Instruments 社の GPIB 通信ボード用ドライバとの併用はできません。
*3: 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。
*4: 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。

⑤MATLAB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">● Microsoft Windows XP Professional● Microsoft Windows XP Home Edition● Microsoft Windows 2000 Professional
MATLAB 対応バージョン	<ul style="list-style-type: none">● MATLAB R14 以上● Data Acquisition Toolbox 2.5 以上
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">● API-AIO (WDM) Ver.1.70 以上

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 販売推進本部

E-mail: promote@contec.jp TEL: 03-5628-0252

■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.jp

FAX: 03-5628-9344

TEL: 03-5628-9286 (弊社営業日9:30~12:00、13:00~17:00)

アナログ入出力 ビギナーズ・ガイドブック Visual Basic2005編 Ver.1.01

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。