KaracriBoard TK0040 ActiveX Control

Visual Basic による KarcriBoard 入門

使用説明書(V1.00)

株式会社 エスアイ創房

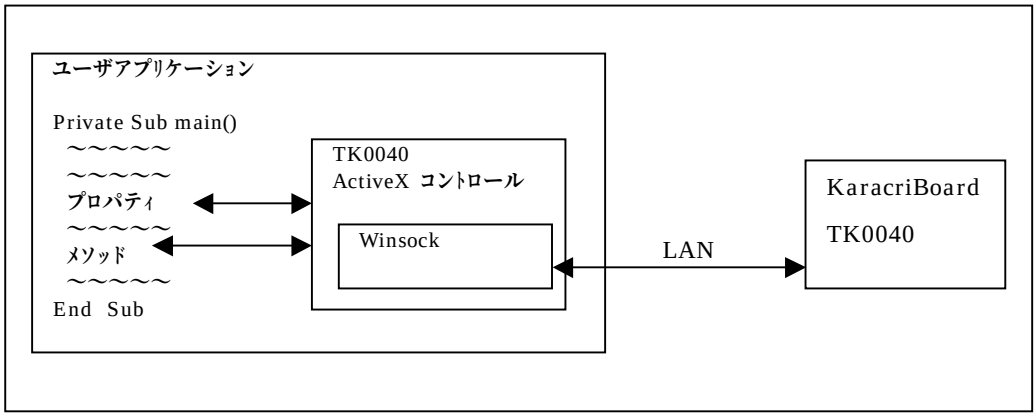
■改定履歴

第 1.00 版　　2007/09/01

目次

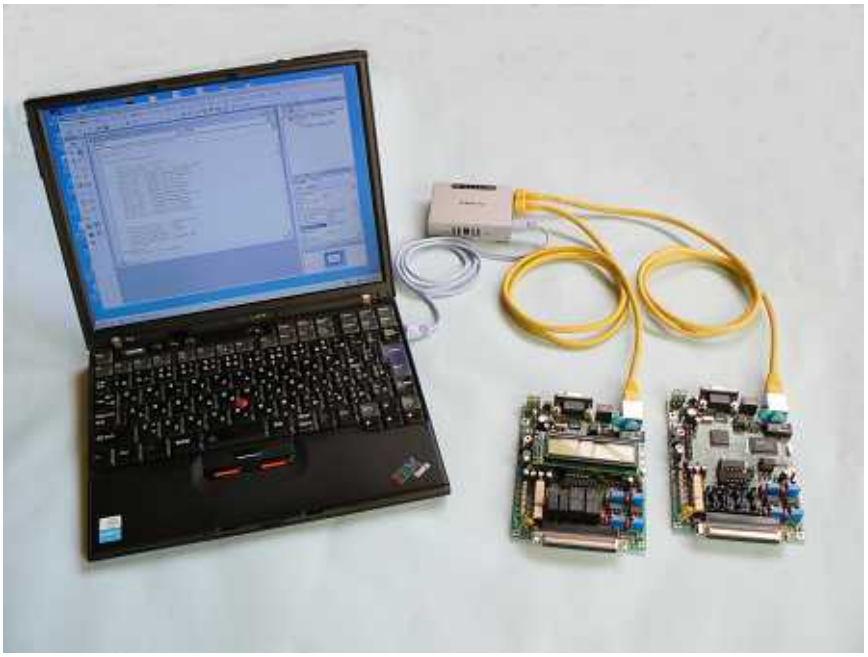
1．概要

TK0040 用のActiveXコントロールを使う事により、VisualBasic から TK0040 をコントロール出来るようになります。また、ユーザアプリケーションからはソケットインターフェイスを使用したプログラミングを行うことなくプロパティ、メソッドを通し、通信コマンドを意識することなく直感的に TK0040 の操作が行なえるようになります。



TK0040　ActiveX コントロール の概要図

＜KarcriBoard と PC の LAN 接続例＞



２．インストール

TK0040 用のActiveXコントロールをダウンロードし、インストールする手順を説明します。

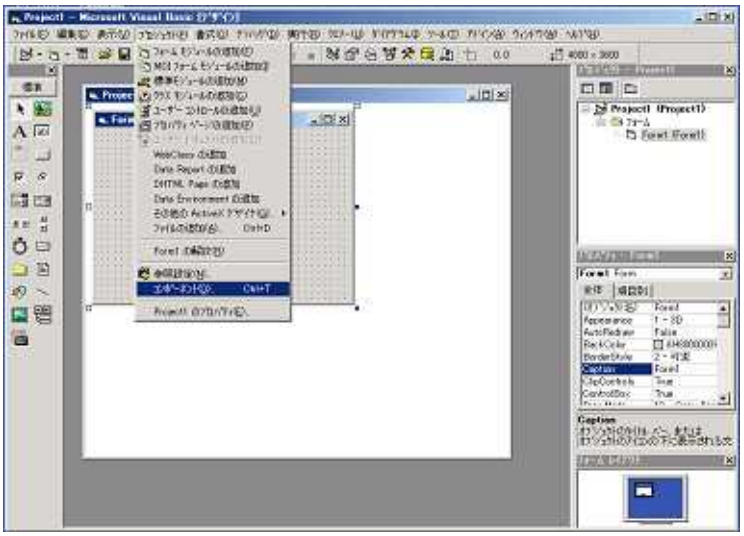
２.１ ダウンロードとセットアップ

1. 古いバージョンのActiveX-OCXを既にご使用の場合には、コントロールパネルの「アプリケーションの追加と削除」でこれに相当する“TK0040 ActiveX Control”を予め削除しておいてください。
2. AvtiveX-OCXファイル(KCX_OCX_TK0040_v???.lzh)を弊社サイトからダウンロードします。(???はバージョン番号を示します)
3. ActiveX-OCXファイルを解凍します。解凍されたフォルダの中には、KCX_OCX_TK0040_v???.EXE (OCXインストーラ) が含まれます。
4. 解凍後、KCX_OCX_TK0040_v???.EXE を実行します。後は画面の指示に従ってください。OCXは、ウインドウズのシステムディレクトリにインストールされます。

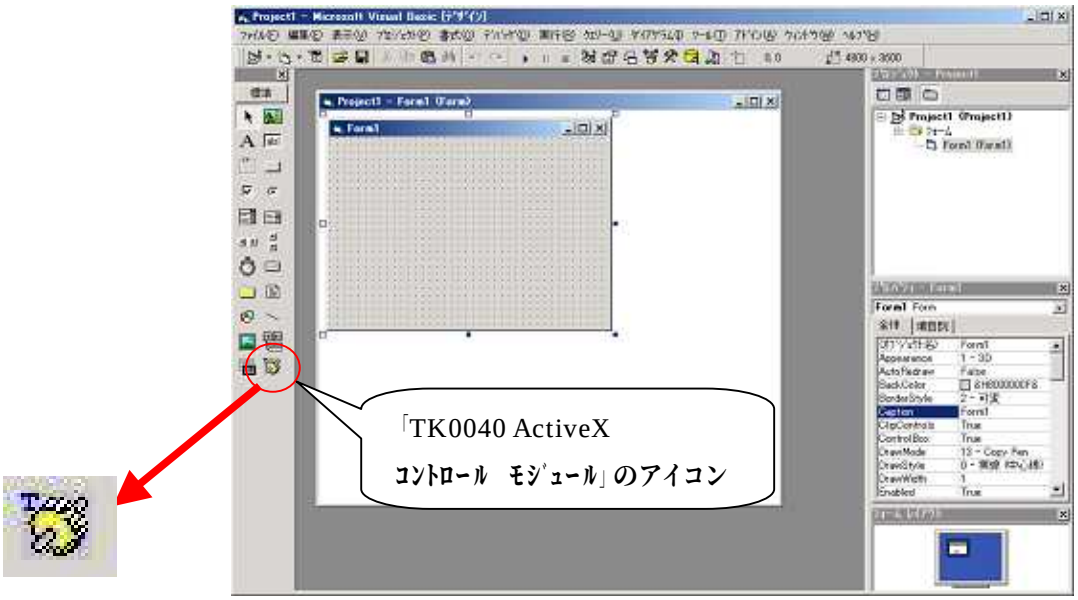
２.２ プロジェクトへ TK0040 ActiveX Control の追加

1. VisualBasicを起動してプロジェクトを一つ作成して下さい。
   VisualBasic を起動すると「新しいプロジェクト」ダイアログが開きますので、“標準.EXE”を選択して「開く(O)」ボタンを選択して下さい。デフォルトフォーム“Form1”を持った画面が表示されます。
   (VisualBasic 起動後に、プロジェクトを作成する場合には、ファイル(F)→プロジェクトの追加(D)で行います。)

2. [プロジェクト(P)]→[コンポーネント(O)]で、コンポーネント･ダイアログを開きます。
   コントロール･タグ上で、"TK0040 ActiveX コントロール モジュール"のチェックを選択(チェック)しOKボタンを押して確定します。ツールボックス上に歯車のアイコン(TK0040)が追加されるのを確認します。




TK0040 ActiveX コントロール モジュール
OK
キャンセル


「TK0040 ActiveX
ｺﾝﾄﾛｰﾙ　ﾓｼﾞｭｰﾙ」のアイコン

3. TK0040 ActiveX Control プログラミング

VisualBasic から、TK0040 用コントロールを使ったいくつかのプログラム例を紹介します。なお、下記の説明で KaracriBoard TK0040 装置に関する設定は、工場出荷時の設定値を使用しています。

※ TK0040 工場出荷時の設定は、以下の通りです。
IP アドレス = 192.168.0.200
コントロールポート = 20000

3.1 TK0040 の装置クロック情報を取得してみる

TK0040 には、装置内の時間を示すCPU実行時間が組み込まれており、装置の時間を簡単に外部から取得することが出来ます。その時間を読み取ってみましょう。

①先ほど作成したプロジェクトに、フォームを1つ作成します。
※通常、“標準.EXE”を選択すると Form1 のデフォルトのフォームが作成されて表示されています。

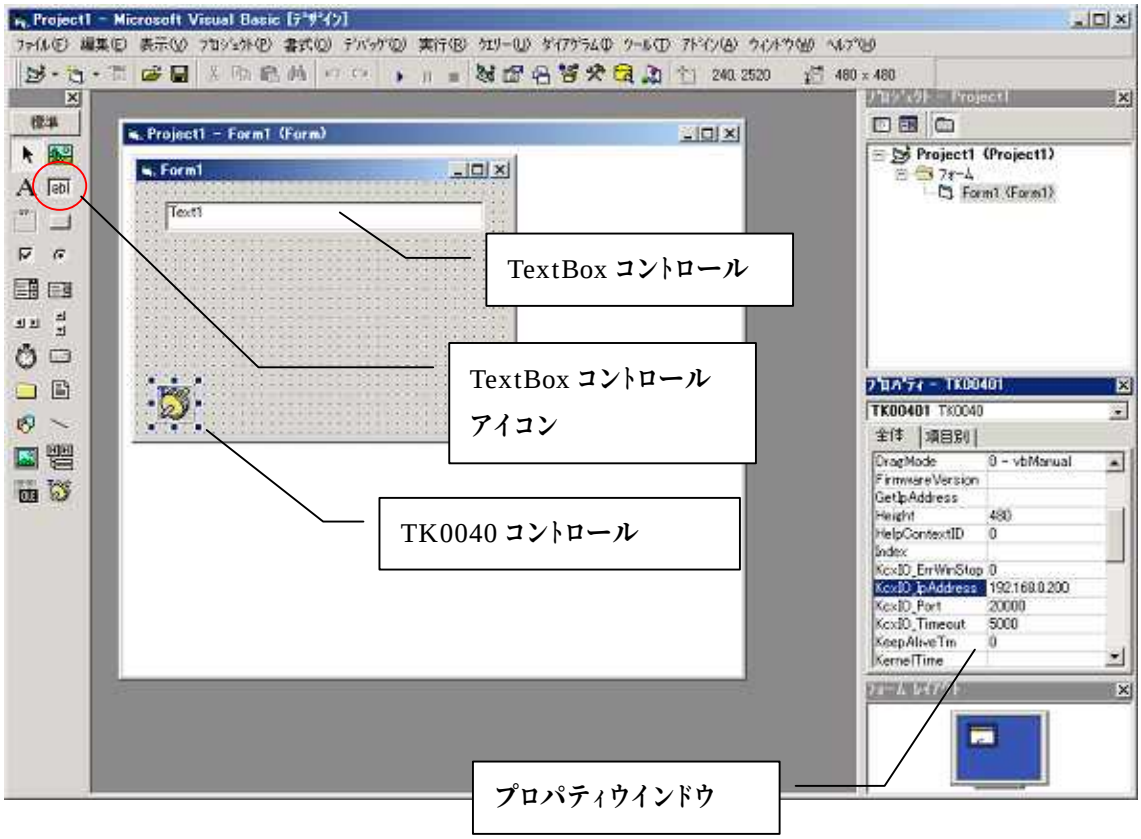
②TK0040 の装置時間を表示する為の、TextBox コントロールを1つ配置します。
オブジェクト名は、”Text”の後ろに自動的に連番が振られ"Text1"となるはずです。
※TextBox コントロールを配置するには、アイコンメニューから TextBox コントロールボタンを
選択して、Form1 上の任意の場所で 2 点クリックすることで行います。

③フォーム上に TK0040 コントロールを1つ配置することでコントロールが使用できるようになります。ツールボックスの TK0040 用コントロール(歯車アイコン)を選択してフォーム上の任意の場所で 2 点クリックして配置して下さい。
オブジェクト名は、通常、コントロール名の末尾に連番が 1 から自動的に付加され"TK00401"となります。
プロパティウィンドウで TK00401 を選択して KcxIO_IpAddress と KcxIO_Port プロパティがデフォルト設定値であることを確認します。
※もし、デフォルト値以外の設定で使用したい場合には、プロパティウインドウで値を変更することが可能です。また、プログラムの先頭（Form_Load()の冒頭）で以下のように記述することも出来ます。

```
TK00401.KcxIO_IpAddress  = 192.168.0.200   ‘ TK0040 の IP アドレス
TK00401.KcxIO_Port       = 20000           ‘ TK0040 のポート番号
```

TK0040 との通信エラータイムアウトは（3～60秒）構内 LAN で使用するため、5秒ということにしておきます。

TK00401.KcxIO_Timeout = 5000 ‘ ミリ秒



④プログラム（プロシージャ）の記述

プログラムは、以下の通りです。

プログラム実行時、つまり、Form_Load() 実行時に、TK00401 オブジェクト の GetMachineInfo メソッドを実行し、TK0040 の装置情報を取得します。通信エラーが発生しなければ、KernelTime プロパティにカーネルタイマカウンタ値が格納されていますので、これをテキストボックスに表示してみます。

Form1 のウインドウ上でダブルクリックするとコードウインドウが表示されますので、以下のようにプログラムを記述して下さい。

```
Private Sub Form_Load()
    TK00401.KcxIO_IpAddress   = 192.168.0.200
    TK00401.KcxIO_Port        = 20000
    TK00401.KcxIO_Timeout     = 5000   ‘ ミリ秒
    TK00401.GetMachineInfo
    Text1.Text = TK00401.KernelTime
End Sub
```

プログラムを記述した後、特に終了するための作業はありません。

プログラムの記述が終わったら“閉じる”ボタンでコードウインドウを閉じて構いません。再度編集したい場合には、フォーム上でダブルクリックして下さい。

⑤実行

メニューの[実行(R)]→[開始(S)]を選択して、プログラムを何度か実行(一度、[終了(E)]を選択してから再度[開始(S)]を選択する)してみてください。

テキストボックスに表示される装置時間(CPU実行時間)が、実行毎に増えていれば、装置及びプログラムは正常に動作しています。

３．２ TK0040 のデジタル＆アナログ入力値を定期的に計測表示してみる

KaracriBoard TK0040 には、6個の接点と4個のアナログ状態を入力をすることができます。その状態を「定期的」に読み取ってみましょう。
ここでは、説明簡略化のため、接点1チャネルとアナログ2チャネル分を対象としました。
TK0040 のアナログ計測レンジは、0～＋5V で分解能は 10bit です。

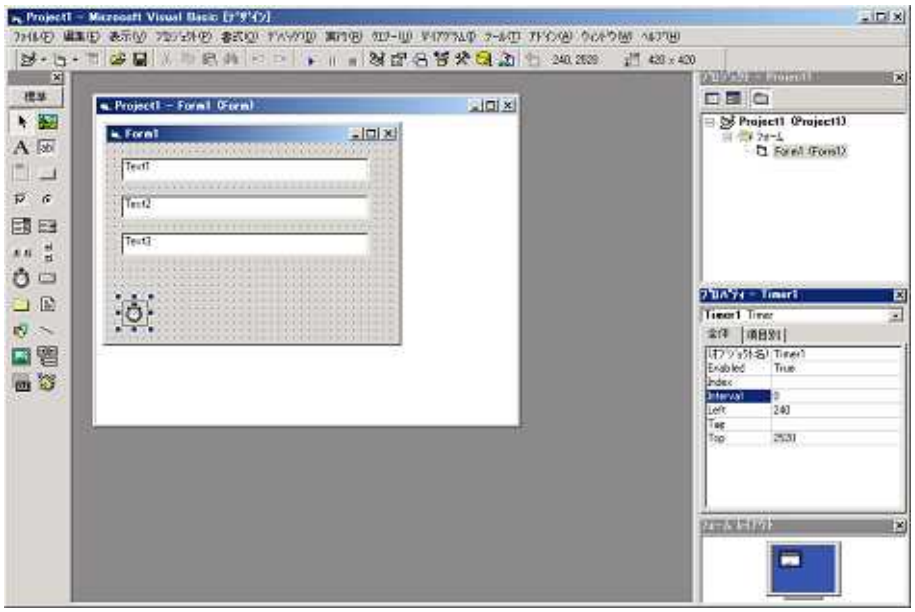
①フォームを1つ作成します。
※“標準.EXE”を選択してプロジェクトを作成すると、通常、Form1 のデフォルトのフォームが自動作成されて表示されています。

②接点1つ、アナログ2つ分の状態を表示する為の、TextBox コントロールを都合3つ配置します。
オブジェクト名は、それぞれ"Text1","Text2","Text3"としました。

③TK0040 を定期的に計測するには、2つの方法があります。一つは、TK0040 装置がもつイベント送信機能を使う方法（3.4参照）と、ポーリング（定周期計測）を行う方法です。

1) ポーリング計測
PC側から TK0040 に定期的にアクセスしてデータを取得する計測方法です。
2) イベント（リアルタイム）計測
TK0040 側で、入出力状態がある範囲を越えて変化した場合などに、そのタイミングでリアルタイムに計測値をPCで取得する方法です。

ここでは、タイマーコントロールを使いインターバル周期を発生させポーリングする方法で計測してみます。
時計のアイコンのタイマーコントロールをクリックしてオブジェクトを1つフォームに配置します。オブジェクト名は、"Timer1"としました。この時、プロパティウィンドウ内の Interval プロパティには、TK0040 をインターバル計測させたい間隔時間をミリ秒単位で設定します。ここでは、1000(1 秒)に設定しています。また、タイマーの Enabled プロパティの初期値には、False※を選択しておきましょう。
（※False にしておく理由：プログラム内でタイマーを有効にしたいタイミングで起動するため）

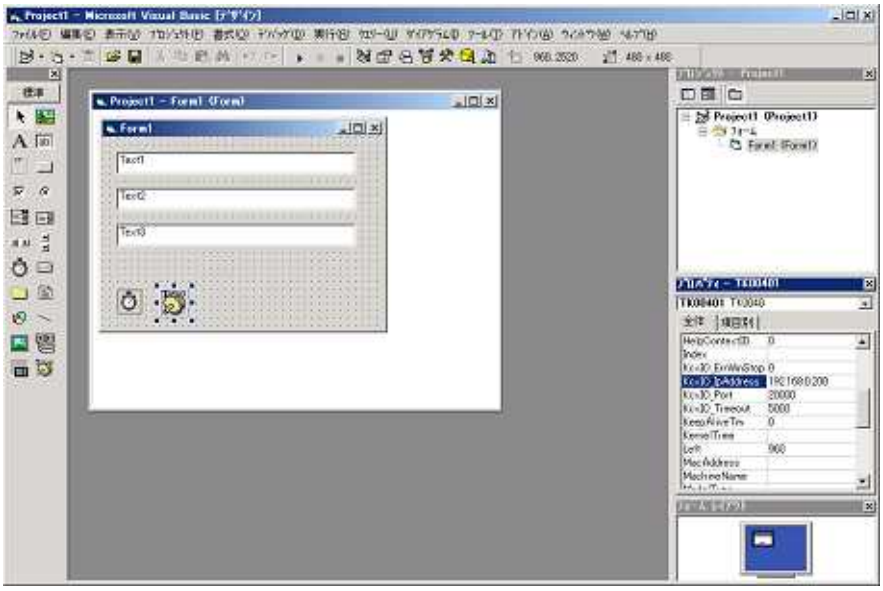


④ツールボックスの TK0040 用コントロールを選択し、TK0040 コントロールを1つ配置します。
オブジェクト名は、"TK00401"となるはずです。
プロパティウィンドウ内の KcxIO_IpAddress と KcxIO_Port プロパティが TK0040 のデフォルト状態と同じであることを確認します。

TK00401.KcxIO_IpAddress = 192.168.0.200
TK00401.KcxIO_Prot = 20000

TK0040 との通信エラータイムアウトは構内 LAN で使用するため、5秒ということにしておきます。

TK00401.KcxIO_Timeout = 5000



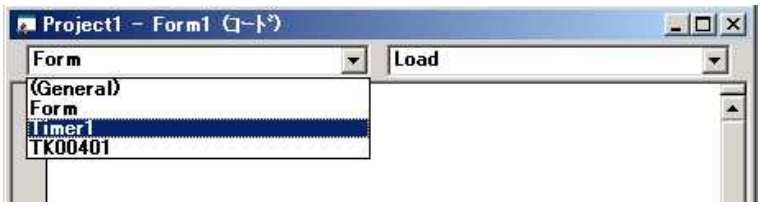
⑤プログラム（プロシージャ）の記述

プログラムは、以下の通りです。

プログラムでは、実行時(Form_Load())に、タイマーの実行を有効にします。

```
Private Sub Form_Load()
    Timer1.Enabled = True   ' 計測インターバルタイマーを有効にする
End Sub
```

次に、タイマーイベントが発生した時に実行されるイベントプロシージャに TK0040 装置と通信してデータを取得する処理を記述します。以下の様に、コードウインドウで **Timer1_Timer** プロシージャを選択してください。イベントプロシージャのコードが追加されます。



Timer1.Interval で設定した時間が経過するとタイマーイベントが発生して Timer1_Timer()が起動され TK0040 の GetIoData メソッドが実行されると、デジタル入力状態と、アナログ入力計測値が、それぞれ、Di_Stat と Ai_Val のプロパティに格納されます。これをテキストボックスに表示します。

```
Private Sub Timer1_Timer()
    TK00401.GetIoData                ' デジタル入力状態、アナログ入力計測値の取得
    Text1.Text = TK00401.Di_Stat(1)   ' 1ch 目のデジタル入力状態表示
    Text2.Text = TK00401.Ai_Val(1)    ' 1ch 目のアナログ入力値表示
    Text3.Text = TK00401.Ai_Val(2)    ' 2ch 目のアナログ入力値表示
End Sub
```

（通信エラーが起きたとき）

TK0040 の GetIoData メソッドが実行されると、TK0040 とネットワーク経由で通信が発生しますが、通信エラーが発生した場合には、デフォルト設定では、GetIoData メソッドが KcxIO_Timeout で設定された時間経過後タイムアウトして復帰します。このとき、次のように GetIoData メソッドの返り値を取得するとエラーの種類を判断することができます。（4.3エラーコード一覧参照）

```
    Dim status As Long
    status =TK00401.GetIoData
```

また、ErrWindowStop プロパティを“1”に設定すると、エラー時に以下のようにダイアログが表示され

プログラムの実行が中断されます。

プログラムのデバッグ時には、エラーダイアログの表示は有効ですが、運用時には、通信エラーが発生しても再度取得するなどの対処を行い計測を継続したいものです。その場合には、ErrWindowStop プロパティを“0”に設定して下さい。

⑥ 実行

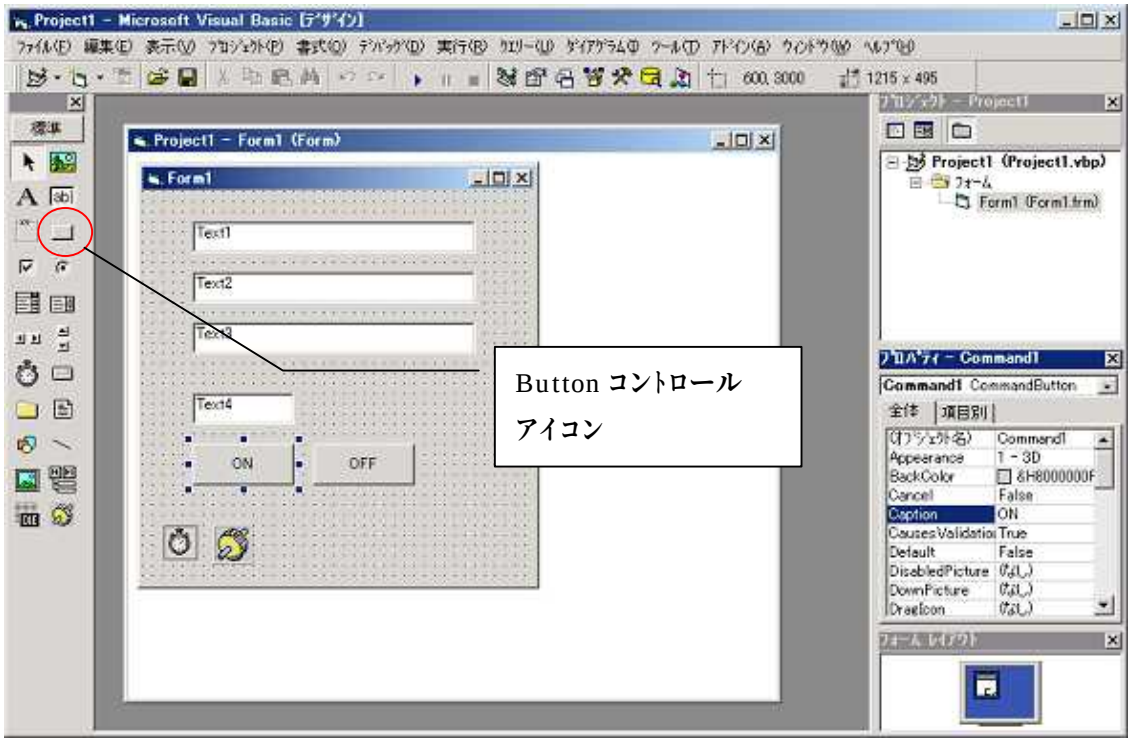
プログラムを実行してみてください。

テキストボックスに、TK0040 装置からの入力データが表示されるはずです。

３．３ TK0040 のリレーを操作してみる

TK0040 には、4つのデジタル出力 (TR 出力又はリレー(オプション)) があり、これを操作することができます。ここでは、リレーが装着されているものとして説明します。
前項の「3.2 TK0040 のデジタル&アナログ値を定期的に計測表示してみる」で作ったものに、デジタル出力を操作するボタンを追加してみます。

①TK0040 のデジタル出力状態を表示させる為の、TextBox コントロールを1つ配置します。
オブジェクト名を、"Text4"としました。
また、リレーを ON/OFF 操作させる為の、Button コントロールを 2つ配置します。
このオブジェクト名は、それぞれ、ON-> "Command1", OFF-> "Command2" としました。



②プログラム (プロシージャ) の記述

プログラムは、以下の通りです。

```
Private Sub Form_Load()
   Timer1.Enabled = True   ' 計測インターバルタイマーを有効にする
End Sub
```

TK0040 の GetIoData メソッドを実行すると、接点状態、アナログ入力の他に現在のデジタル出力状態も同時に取得できる仕様になっています。デジタル出力の状態は、Do_Stat プロパティに格納されていますので、これをテキストボックスに表示します。

```
Private Sub Timer1_Timer()
   Dim status As Long   ' メソッドの返り値
   status = TK00401.GetIoData          ' デジタル入出力状態、アナログ入力計測値の取得
    If (status > 0) Then
       Text1.Text = TK00401.Di_Stat(1)    ' 1ch 目のデジタル入力状態表示
       Text2.Text = TK00401.Ai_Val(1)     ' 1ch 目のアナログ入力値表示
       Text3.Text = TK00401.Ai_Val(2)     ' 2ch 目のアナログ入力値表示
       Text4.Text = TK00401.Do_Stat(1)    ' 1ch 目のリレー状態表示
   End If
End Sub
```

フォームに配置した、ON ボタンを押すと、Command1_Click() が実行され、TK0040 オブジェクトの SetDout メソッドを実行し 1ch 目のリレー出力 ON が送信されます。
Command2_Click() の場合は同様に 1ch 目のリレー出力 OFF が送信されます。
SetDout メソッドの引数に DO_ON を指定すると、リレーはオン(接点短絡)、DO_OFF を指定すると、オフ(接点開放)を要求することになります。操作要求しない、つまり現状のままでよいという場合には、DO_NOCHANGE を指定します。また、SetDout メソッドを実行すると、Do_Stat プロパティに反映されますので、操作した後のリレーの状態を即時にテキストボックスに表示するように記述します。
尚、DO_ON、DO_OFF、DO_NOCHANGE は、それぞれ、1、0、-1 に内部定義されています。

```
Private Sub Command1_Click()
   ' 1ch 目のリレーON を送信
   TK00401.SetDout   DO_ON, DO_NOCHANGE, DO_NOCHANGE, DO_NOCHANGE   } どちらの記述でもOK
   TK00401.SetDout   1, -1, -1, -1                                  }
   Text4.Text = TK00401.Do_Stat(1)    ' 1ch 目のリレー状態表示
End Sub

Private Sub Command2_Click()
   ' 1ch 目のリレーOFF を送信
   TK00401.SetDout DO_OFF, DO_NOCHANGE, DO_NOCHANGE, DO_NOCHANGE    } どちらの記述でもOK
   TK00401.SetDout 0, -1, -1, -1                                    }
   Text4.Text = TK00401.Do_Stat(1)    ' 1ch 目のリレー状態表示
End Sub
```

③実行

プログラムを実行してみてください。

テキストボックスに TK0040 の接点入力状態、アナログ入力値を表示しながら、デジタル出力の ON/OFF 操作が出来るはずです。

３．４ TK0040 のイベント機能の使用法

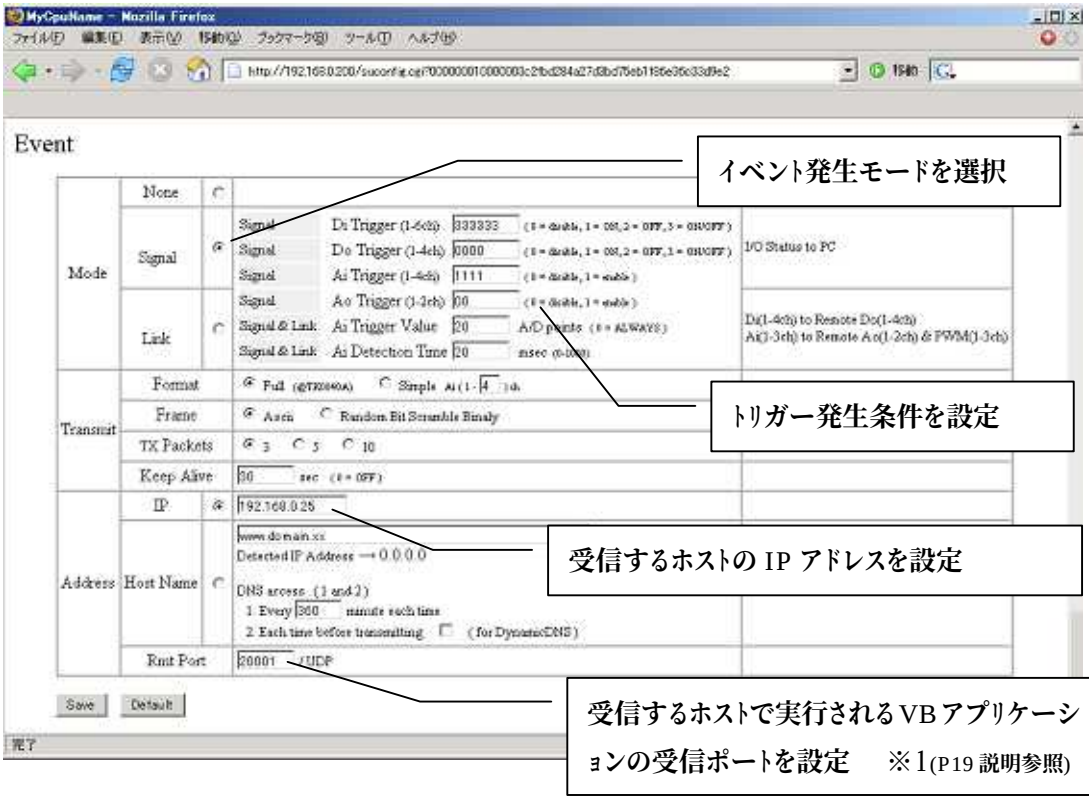
ここまで、TK0040 を定期的に計測するためにタイマーコントロールを使用してポーリングアクセスを行ってきましたが、TK0040 装置自身のもつイベント送信機能を使う方法を解説します。

①TK0040 の Evnet 機能を設定する

イベント送信機能を使用するためには、はじめに TK0040 のシステム設定でイベント送信タイミングの設定を行います。(TK0040 取扱説明書を参照)

TK0040 には、いくつかのイベント発生トリガーを設定できますが、ここでは、接点入力状態やアナログ入力値の変化によりイベントを発生させて、VB アプリケーションで受信してみましょう。TK0040 のシステム設定の Event 設定画面で、以下のように設定して下さい。

(設定後、Save ボタンを押して、TK0040 装置の再起動を行い、再度、設定を確認して下さい。)



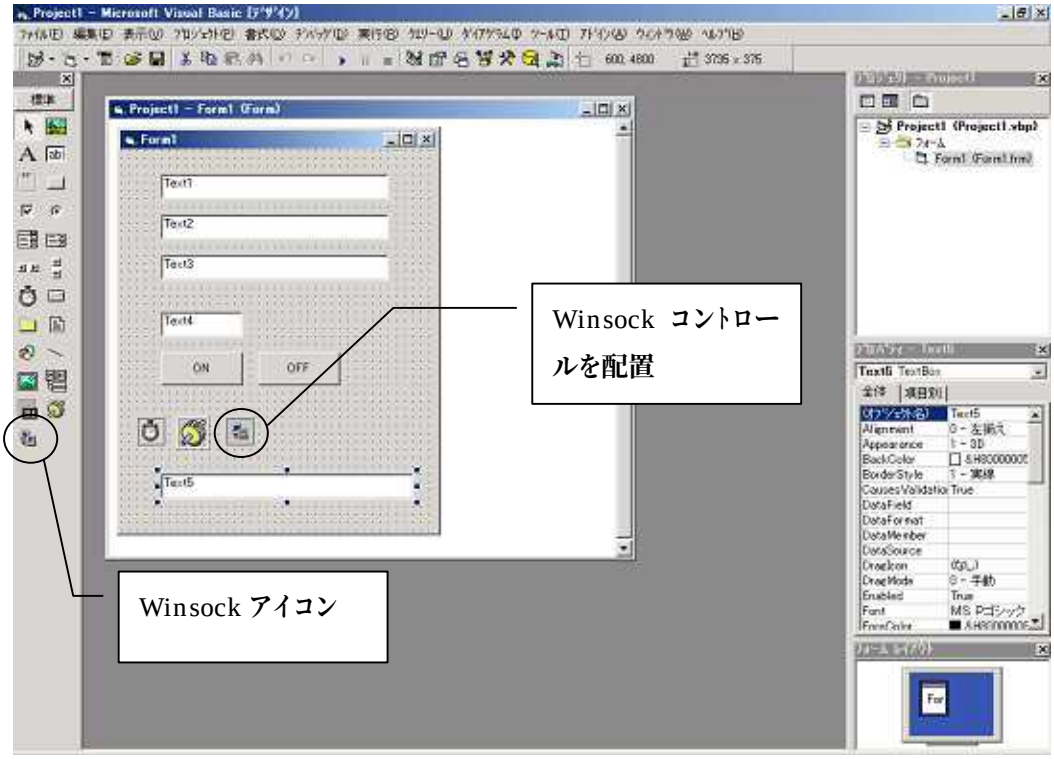
IP (イベント送信先ホストの IP アドレス)： 192.168.0.25

Remot Port (VB アプリケーションの待受けポート)： 20001

②Winsock コントロールの追加

次に、VB 側の準備としてプロジェクトのコンポーネントメニューから Winsock コントロールをチェックして、プロジェクトに Winsock コントロールを追加してください。

コントロールメニューに Winsock アイコンが追加されますので、クリックしてフォームに追加して配置します。さらに、受信データ表示用として TextBox コントロールを Text5 として配置しておきます。



③プロシージャの記述

プログラムは、以下の通りです。
TK0040 のイベント機能を使用しますので、実行時(Form_Load())に、タイマーコントロールの実行を無効にします。
次に、Winsock コントロールの初期化を行います。

```
Private Sub Form_Load()
    Timer1.Enabled = False   ' 計測インターバルタイマーを無効にしておく
    (省略)
   Winsock1.Protocol = sckUDPProtocol
   Winsock1.Bind 20001
   Winsock1.RemoteHost = "192.168.0.200"
   Winsock1.RemoteHost =  "INADDR_ANY"
End Sub
```

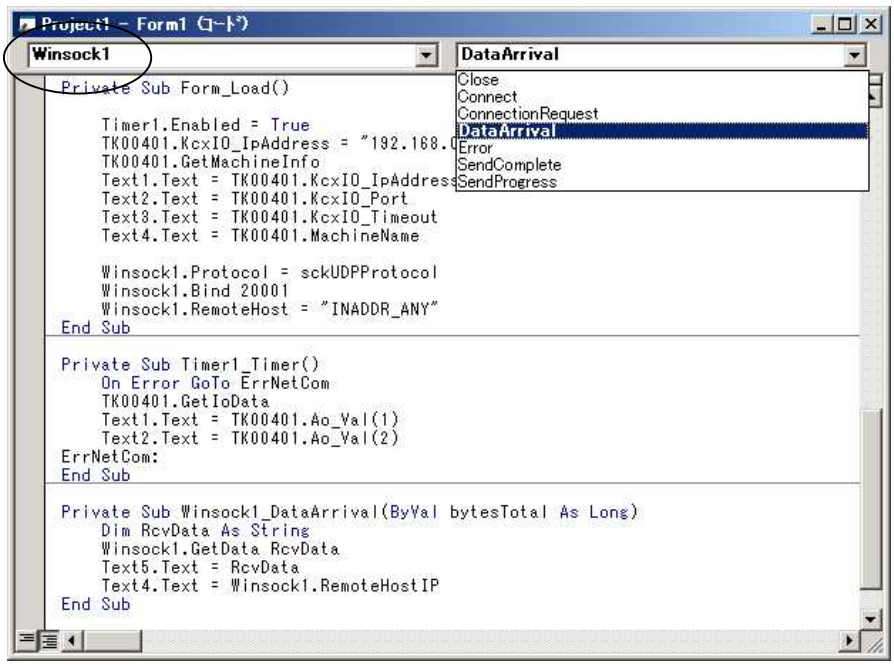
プロトコルを UDP 通信に設定

PC側プログラムの待受けポートをバインド ※2

イベント受信する TK0040 装置の IP アドレスを指定する場合

イベント受信する対象を全ての装置にする場合 (通常こちらが良いでしょう)

(※1(P17)、※2の値が同じであることを確認して下さい。)
これで、イベント受信の準備ができました。Winsock コントロールは、データを受信すると DataArrival 関数が呼び出される仕様になっています。以下のように、コードウィンドウで Winsock1 オブジェクトを選択して DataArrival 関数を選択すると Winsock1_DataArrival 関数が追加されます。



データをTK0040から受信しますと、Winsock1_DataArrival関数がコールされますので、以下のように記述することで、受信データを Text5 のテキストボックスに表示することができます。

```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
    Dim RcvData As String
    Winsock1.GetData RcvData
    Text5.Text = RcvData
    Text4.Text = Winsock1.RemoteHostIP
End Sub
```

Dim 変数名 As String
ここでは、RcvData が変数名です。

受信したデータ文字列

受信した TK0040 装置の IP アドレス

③実行

プログラムを実行してみてください。

ﾄﾘｶﾞｰ発生条件に合う接点入力をＯＮ／ＯＦＦしたり、アナログ入力値を変化させてみてください。ﾘｱﾙﾀｲﾑでテキストボックスに TK0040 から受信したデータが表示出来るはずです。

（※もし、うまく受信できない場合は、受信ホスト側で、指定したポート番号がウイルス対策ソフトやファイアフォールソフトで制限されていないか確認して下さい。）

4. KaracriBoard TK0040 ActiveX Control リファレンス

4.1 メソッド

4.1.1 メソッド一覧

下記のメソッド一覧では、各メソッドを実行することにより取得できるプロパティの対応を示します。

項番	名称	取得／設定プロパティ	説明
4.1.2	GetMachineInfo	ModelType FirmwareVersion MachineName MacAddress、GetIpAddress StartUpMode、KernelTime	装置情報の取得
4.1.3	GetIoData	Di_Stat、Di_TmStat Di_Count、Do_Stat Ai_Val、Ao_Val、PWMo_Val Msg1、KernelTime	接点入力(Di)、リレー出力(Do)、アナログ入力(Ai)全チャンネルの状態を取得
4.1.4	GetDiHoldTm	Di_TmVal	接点入力(Di)各チャンネルの ON 保持状態
4.1.5	GetDioEventTrig	Di_EvtTrig、Do_EvtTrig	接点入力(Di)、リレー出力(Do)状態値変化時のイベント発生条件を取得
4.1.6	GetAioEventTrig	Ai_EvtTrig、Ao_EvtTrig Ai_EvtJudgeVal Ai_EvtDetecTm	アナログ入力(Ai)、出力(Ao)値変化時のイベント発生条件を取得
4.1.7	GetMsg1	Msg1	メッセージ 1 を取得
4.1.8	GetMsg2	Msg2	メッセージ 2 を取得
4.1.9	GetKeepAliveTm	KeepAliveTm	KeepAlive イベント発生時間を取得
4.1.10	SetDout	Do_Stat	リレー出力(Do)各チャンネルに状態を設定
4.1.11	SetAout	Ao_Val	アナログ出力(Ao)を操作
4.1.12	SetPWMout	PWMo_Val	PWM 出力(CH1、CH2、CH3)を操作
4.1.13	SetDiCount	Di_Count	接点入力(Di)の開閉カウント値の初期化
4.1.14	SetDiCountALL0	Di_Count	接点入力(Di)の開閉カウント値の全チャンネル初期化
4.1.15	SetDioEventTrig	Di_EvtTrig、Do_EvtTrig	接点入力(Di)、リレー出力(Do)状態値変化時のイベント発生条件を設定
4.1.16	SetAioEventTrig	Ai_EvtTrig、Ao_EvtTrig	アナログ入力(Ai)、出力(Ao)値変化時のイベント発生条件を設定
4.1.17	SetMsg1	Msg1	メッセージ 1 を設定
4.1.18	SetMsg2	Msg2	メッセージ 2 を設定
4.1.19	SetKeepAliveTm	KeepAliveTm	KeepAlive イベント発生時間を設定

４.１.２ GetMachineInfo メソッド

装置のシステム情報を取得します。

構文

object. GetMachineInfo

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

戻り値 長整数型 (Long)

値	内容
正値 (≧0)	正常終了 (取得データ長)
負値 (<0)	エラー (エラーコード一覧参照)

解説

各メソッド実行前の初期化処理として、KcxIO_IpAddress プロパティ、KcxIO_Port プロパティの設定が必要です。設定を省略した場合は、デフォルト値*1 で指定された装置に接続を試みます。デフォルト値以外に設定した装置と通信する場合には必ず設定して下さい。

*1 デフォルト値

KcxIO_IpAddress のデフォルト値 → 192.168.0.200

KcxIO_Port のデフォルト値 → 20000

(例) IP アドレス：192.168.0.50、ポート番号：30001 で通信する場合

```
TK00401.KcxIO_IpAddress  = 192.168.0.50
TK00401.KcxIO_Port       = 30001
TK00401.GetMachineInfo
```

４.１.３ GetIoData メソッド

アナログ入出力、デジタル入出力の全チャンネルの状態を取得します。

構文

object. GetIoData

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

戻り値 長整数型 (Long)

値	内容
正値 (≧0)	正常終了 (取得データ長)
負値 (<0)	エラー (エラーコード一覧参照)

解説

本メソッド実行後、Di_Stat、Di_TmStat、Di_Count、Do_Stat、Di_Stat、Ai_Val、Ao_Val、PWMo_Val プロパティで各チャンネルの状態を取得します。
戻り値がエラーの場合は、プロパティの値は変更されません。

アナログ各チャンネルの値は、0～1023の長整数値として取得することができます。
デジタル各チャンネルの状態値が、0の場合は、OFF (端子オープン)、1の場合は、ON (端子ショート) を示します。

(例)

```
TK00401.GetIoData
Text1.Text = TK00401.Di_Stat(1)
Text2.Text = TK00401.Do_Stat(1)
Text3.Text = TK00401.Ai_Val(1)
```

４.１.４ GetDiHoldTm メソッド

接点入力(Di)各チャンネルの瞬間 ON 保持状態(カウントダウン値)を取得します。

構文

object. GetDiHoldTm

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

戻り値 長整数型(Long)

値	内容
正値(≧0)	正常終了(取得データ長)
負値(<0)	エラー(エラーコード一覧参照)

解説

本メソッド実行後、Di_TmVal プロパティで各チャンネルの接点入力(Di)の瞬間 ON 保持状態値を取得します。

戻り値がエラーの場合は、プロパティの値は変更されません。

状態値は、接点入力が、

OFF の場合 → 0

ON の場合 → 瞬間 ON 保持時間[システム設定値](秒) × 10

の値として取得できます。

接点入力が ON から OFF に変化すると、状態値は、瞬間 ON 保持時間(秒) × 10 の値から、0.1 秒毎に 1 ずつダウンカウントされ、最後に 0 となり停止します。

瞬間 ON 保持時間が、3 秒の場合、30、29、28、… 2、1、0(停止)と変化していきます。

※瞬間 ON 保持時間に関しては、取扱説明書の解説を参照してください。

(例)

```
TK00401.GetDiHoldTm
Text1.Text = TK00401.Di_TmVal(1)
```

４.１.５ GetDioEventTrig メソッド

接点入力(Di)、デジタル出力(Do)値が変化した時のイベント発生条件を取得します。

構文

object. GetDioEventTrig

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

戻り値 長整数型(Long)

値	内容
正値(≧0)	正常終了(取得データ長)
負値(<0)	エラー(エラーコード一覧参照)

解説

本メソッド実行後、**Di_EvtTrig**、**Do_EvtTrig** プロパティで各チャンネルのイベント発生条件を取得します。

戻り値がエラーの場合は、プロパティの値は変更されません。

(例)

```
TK00401.GetDioEventTrig
Text1.Text = TK00401.Di_EvtTrig(1)
Text2.Text = TK00401.Do_EvtTrig(1)
```

４.１.６ GetAioEventTrig メソッド

アナログ入力（Ai）、アナログ出力(Ao)値が変化した時のイベント発生条件を取得します。

構文

object. GetAioEventTrig

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

戻り値 長整数型（Long）

値	内容
正値（≧0）	正常終了（取得データ長）
負値（<0）	エラー（エラーコード一覧参照）

解説

本メソッド実行後、**Ai_EvtTrig**、**Ao_EvtTrig** プロパティで各チャンネルのイベント発生条件を取得します。また、**Ai_EvtJudgeVal**、**Ai_EvtDetecTm** プロパティは、各チャンネル共通で適用されます。

戻り値がエラーの場合は、プロパティの値は変更されません。

(例)

```
TK00401.GetAioEventTrig
Text1.Text = TK00401.Ai_EvtTrig(1)
Text2.Text = TK00401.Ao_EvtTrig(1)
Text3.Text = TK00401.Ai_EvtJudgeVal
Text4.Text = TK00401.Ai_EvtDetecTm
```

４.１.７ GetMsg1 メソッド

Msg1 に設定されているメッセージ文字列を取得します。

構文

object. **GetMsg1**

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

戻り値 長整数型 (Long)

値	内容
正値 (≧0)	正常終了 (取得データ長)
負値 (<0)	エラー (エラーコード一覧参照)

解説

本メソッド実行後、**Msg1** プロパティで設定メッセージを取得します。

戻り値がエラーの場合は、プロパティの値は変更されません。

(例)

```
TK00401.GetMsg1
Text1.Text = TK00401.Msg1
```

４.１.８ GetMsg2 メソッド

Msg2 に設定されているメッセージ文字列を取得します。

構文

object. **GetMsg2**

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

戻り値 長整数型 (Long)

値	内容
正値 (≧0)	正常終了 (取得データ長)
負値 (<0)	エラー (エラーコード一覧参照)

解説

本メソッド実行後、**Msg2** プロパティで設定メッセージを取得します。

戻り値がエラーの場合は、プロパティの値は変更されません。

(例)

```
TK00401.GetMsg2
Text1.Text = TK00401.Msg2
```

４.１.９ GetKeepAliveTm メソッド

KeepAlive イベントの発生インターバル時間(秒)を取得します。

構文

object. **GetKeepAliveTm**

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

戻り値 長整数型 (Long)

値	内容
正値 (≧0)	正常終了 (取得データ長)
負値 (<0)	エラー (エラーコード一覧参照)

解説

本メソッド実行後、**KeepAliveTm** プロパティで値を取得します。

戻り値がエラーの場合は、プロパティの値は変更されません。

(例)

```
TK00401.GetKeepAliveTm
Text1.Text = TK00401.KeepAliveTm
```

４.１.１０ SetDout メソッド

デジタル出力(Do)各チャンネルに状態を設定します。

構文

object. **SetDout**　d1, d2, d3, d4

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
d1 ~ d4	Long	デジタル出力(1～4CH)各チャンネルの設定値

設定値

引数 d1,d2,d3,d4 の設定値

定　数	値	説　明
DO_OFF	0	OFF(デジタル出力オープン)
DO_ON	1	ON(デジタル出力ショート)
DO_NOCHANGE	-1(0,1 以外)	無変更

戻り値

値	内容
正値(≧0)	正常終了(取得データ長)
負値(<0)	エラー(エラーコード一覧参照)

解説

設定した状態は **Do_Stat** プロパティに反映されます。

(コマンド送信後、OCX 内部で GetIoData を呼び出してプロパティにセットしています。)

設定値が(0,1)以外の場合、そのチャンネルの状態は変更されません。

(例)

TK00401.SetDout　ON, OFF, OFF, DO_NOCHANGE

TK00401.SetDout　1, 0, 0, -1

(CH1:ON、CH2:OFF、CH3:OFF、CH4:無変更)

４.１.１１ SetAout メソッド

アナログ出力(Ao)各チャンネルに値を設定します。

構文

object. **SetAout** a1, a2

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
a1	Long	チャンネル１の設定値
a2	Long	チャンネル２の設定値

設定値

引数 a1、a2 の設定値

値	説　明
0～255	DA コンバータ値
-1	無変更(現状維持)

戻り値 長整数型(Long)

値	内容
正値(≧0)	正常終了(取得データ長)
負値(<0)	エラー(エラーコード一覧参照)

解説

本メソッド実行後、**Ao_Val** プロパティの値に反映されます。

(コマンド送信後、OCX 内部で GetIoData を呼び出してプロパティにセットしています。)

設定値が(0～255)以外の場合、そのチャンネルの状態は変更されません。

(例)

```
TK00401.SetAout   128, 255
TK00401.SetAout   77, -1
                  (CH1:77、CH2:無変更)
```

４.１.１２ SetPWMout メソッド

PWM 出力 (Ch1、Ch2、Ch3) 値を設定します。

構文

object. **SetPWMout** a1, a2, a3

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
a1	Long	チャンネル 1 の設定値
a2	Long	チャンネル 2 の設定値
a3	Long	チャンネル 3 の設定値

設定値

引数 a1、a2、a3 の設定値

値	説　明
0～10000	PWM スケール値
-1	無変更 (現状維持)

戻り値 長整数型 (Long)

値	内容
正値 (≧0)	正常終了 (取得データ長)
負値 (<0)	エラー (エラーコード一覧参照)

解説

本メソッド実行後、**PWMo_Val** プロパティの値に反映されます。

(コマンド送信後、OCX 内部で GetIoData を呼び出してプロパティにセットしています。)

設定値が(0～10000)以外の場合、そのチャンネルの状態は変更されません。

(例)

TK00401. SetPWMout 1000, -1, 3000

(CH1:1000,CH2:無変更,CH3:3000)

４.１.１３ SetDiCount メソッド

接点(Di)開閉カウント値を指定の値で初期化します。

構文

object. **SetDiCount** d1, d2

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
d1	Long	接点入力(Di)チャンネルの指定
d2	Long	指定チャンネルのカウンタ設定値

設定値

引数 d1 の設定値

値	システム設定(Web 画面：DiOnCounter)が以下の設定の場合
1～6	Count
1～6	Count&Rom Memory
3～6	RealTime HW Count

引数 d2 の設定値

値	システム設定(Web 画面：DiOnCounter)が以下の設定の場合
0～999999999	Count
0～999999999	Count&Rom Memory
0～65535	RealTime HW Count

戻り値 長整数型(Long)

値	内容
正値(≧0)	正常終了(取得データ長)
負値(<0)	エラー(エラーコード一覧参照)

解説

本メソッド実行後、**Di_Count** プロパティの値に反映されます。

(コマンド送信後、OCX 内部で GetIoData を呼び出してプロパティにセットしています。)

設定した値が上記設定値の範囲外の場合、そのチャンネルの状態は変更されません。

(例)

TK00401. SetDiCount　1, 999

(CH1 を 999 に設定)

４.１.１４ SetDiCountALL0 メソッド

接点(Di)開閉カウント値を全チャンネル 0 で初期化します。

構文

object. SetDiCountALL0

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

戻り値 長整数型(Long)

値	内容
正値(≧0)	正常終了(取得データ長)
負値(<0)	エラー(エラーコード一覧参照)

解説

本メソッド実行後、**Di_Count** プロパティの値に反映されます。

(コマンド送信後、OCX 内部で GetIoData を呼び出してプロパティにセットしています。)

(例)

TK00401. SetDiCountALL0

(全チャンネルを 0 に設定)

４.１.１５ SetDioEventTrig メソッド

接点入力(Di)、リレー出力(Do)状態値変化時のイベント発生条件を設定します。

構文

object. **SetDioEventTrig** d1, d2

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
d1	String	Di(1-6Ch)イベント検出モード
d2	String	Do(1-4Ch)イベント検出モード

設定値

引数 d1 の設定値(6 桁必須)

値	説　明
0	無検出
1	ON 時に検出
2	OFF 時に検出
3	ON/OFF 時に検出
-	無変更(現状維持)

引数 d2 の設定値(4 桁必須)

値	説　明
0	無検出
1	ON 時に検出
2	OFF 時に検出
3	ON/OFF 時に検出
-	無変更(現状維持)

戻り値 長整数型(Long)

値	内容
正値(≧0)	正常終了(取得データ長)
負値(<0)	エラー(エラーコード一覧参照)

解説

本メソッド実行後、**Di_EventTrig** プロパティ、**Do_EventTrig** プロパティの値に反映されます。

(コマンド送信後、OCX 内部で GetDioEventTrig を実行してプロパティにセットしています。)

(例)

TK00401.SetDioEventTrig　“112200”, “11-0”

Di-> (CH1-2:ON 時、CH3-4:OFF 時、CH5-6:無検出)

Do-> (CH1-2:ON 時、CH3:無変更、CH4: OFF 時)

４.１.１６ SetAioEventTrig メソッド

アナログ入力(Ai)、アナログ出力(Ao)値変化時のイベント発生条件を設定します。

構文

object. **SetAioEventTrig** a1, a2, a3, a4

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
a1	Long	Ai 検出値差(検出判定変化値)
a2	Long	Ai 検出間隔(検出インターバル時間(msec))
a3	String	Ai(1-4Ch)イベント検出モード
a4	String	Ao(1-2Ch)イベント検出モード

設定値

引数 a1 の設定値

値	説　明
0～9999	0で常時イベント発生、1024以上では イベントは発生しません。

引数 a2 の設定値

値	説　明
0～1000	ミリ秒

引数 a3,a4 の設定値

値	説　明
0	無検出
1	検出
-	無変更(現状維持)

戻り値 長整数型(Long)

値	内容
正値(≧0)	正常終了(取得データ長)
負値(<0)	エラー(エラーコード一覧参照)

解説

本メソッド実行後、**Ai_EventTrig**、**Ao_EventTrig**、**Ai_EvtJudgeVal**、**Ai_EvtDetecTm** の各プロパティの値に反映されます。

(コマンド送信後、OCX 内部で GetAioEventTrig を実行してプロパティにセットしています。)

(例)

TK00401.SetAioEventTrig　20, 500, “100-”, “01”

(Ai 検出値差:20、Ai 検出間隔:500msec、

Ai(Ch1:検出,Ch2-3:無検出,Ch4:無変更)、Ao(Ch1:無検出,Ch2:検出)

４.１.１７ SetMsg1 メソッド

共有メッセージ 1 に文字列を設定します。

構文

object. **SetMsg1**　s1

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
s1	String	メッセージ 1 文字列

設定値

引数 s1 の設定値

値	説　明
40 桁以下の文字列	空白を挟まない連続した文字列を指定します。但し、NULL と NULLCLEAR という文字列は特別な意味で予約されています。 NULL は、無効であることを意味する使用上無意味なものです。 NULLCLEAR は、設定されているメッセージをクリアする時に使用する文字列です。

戻り値

値	内容
正値 (≧0)	正常終了 (取得データ長)
負値 (<0)	エラー (エラーコード一覧参照)

解説

本メソッド実行後、**Msg1** プロパティの値に反映されます。

(コマンド送信後、OCX 内部で GetMsg1 を実行してプロパティにセットしています。)

(例 1)　TK00401.SetMsg1　123-abc-ABC

(例 2)　TK00401.SetMsg1　NULLCLEAR

(メッセージをクリア)

４.１.１８ SetMsg2 メソッド

共有メッセージ 2 に文字列を設定します。

構文

object. **SetMsg2**　s1

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
s1	String	メッセージ 2 文字列

設定値

引数 s1 の設定値

値	説　明
40 桁以下の文字列	空白を挟まない連続した文字列を指定します。但し、NULL と NULLCLEAR という文字列は特別な意味で予約されています。 NULL は、無効であることを意味する使用上無意味なものです。 NULLCLEAR は、設定されているメッセージをクリアする時に使用する文字列です。

戻り値

値	内容
正値 (≧0)	正常終了 (取得データ長)
負値 (<0)	エラー (エラーコード一覧参照)

解説

本メソッド実行後、**Msg2** プロパティの値に反映されます。

(コマンド送信後、OCX 内部で GetMsg2 を実行してプロパティにセットしています。)

(例 1)　TK00401.SetMsg2　456-def-DEF

(例 2)　TK00401.SetMsg2　NULLCLEAR

(メッセージをクリア)

４.１.１９ SetKeepAliveTm メソッド

KeepAlive イベント発生時間（秒）を設定します。

構文

object. **SetKeepAliveTm** d1

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
d1	Long	KeepAlive イベント発生時間（秒）

設定値

引数 d1 の設定値

値	説　明
0～9999	0 を設定すると KeepAlive の機能が無効になります。

戻り値

値	内容
正値（≧0）	正常終了（取得データ長）
負値（<0）	エラー（エラーコード一覧参照）

解説

本メソッド実行後、**KeepAliveTm** プロパティの値に反映されます。

（コマンド送信後、OCX 内部で GetKeepAliveTm を実行してプロパティにセットしています。）

(例)

TK00401. SetKeepAliveTm 120

(120 秒に設定)

4.2 プロパティ

4.2.1 プロパティー覧

項番	名 称	データ型	属性	説 明	初期値
4.2.2	ModelType	String	R	型式名称	
4.2.3	FirmwareVersion	String	R	本機のファームウェアバージョン	
4.2.4	MachineName	String	R	機械名称	
4.2.5	MacAddress	String	R	TK0040 の取得 MAC アドレス	
4.2.6	GetIpAddress	String	R	TK0040 の取得 IP アドレス	
4.2.7	StartUpMode	String	R	本機の起動状態	
4.2.8	KernelTime	Long	R	TK0040 のカーネルタイムカウンタ	
4.2.9	KcxIO_Timeout	Long	RW	各メソッド実行時のタイムアウト時間 (ms)	5000
4.2.10	KcxIO_IpAddress	String	RW	送信する TK0040 の IP アドレス (ホスト名)	192.168.0.200
4.2.11	KcxIO_Port	Long	RW	送信する TK0040 のポート番号	20000
4.2.12	MyPC_Port	Long	RW	ローカルコンピュータのポート番号	0
4.2.13	Di_Stat	Long	R	接点入力 (Di) 各チャンネルの状態	
4.2.14	Di_TmStat	Long	R	接点入力 (Di) 各チャンネルの保持状態	
4.2.15	Di_TmVal	Long	R	接点入力 (Di) 各チャンネルの保持状態 (ダウンカウント値)	
4.2.16	Di_Count	Long	R	接点入力 (Di) 各チャンネルの開閉回数	
4.2.17	Do_Stat	Long	R	デジタル出力 (Do) 各チャンネルの状態	
4.2.18	Ai_Val	Long	R	アナログ入力 (Ai) 各チャンネルの状態	
4.2.19	Ao_Val	Long	R	アナログ出力 (Ao) 各チャンネルの状態	
4.2.20	PWMo_Val	Long	R	PWM 出力スケール値	
4.2.21	Di_EvtTrig	Long	R	接点入力 (Di) 各チャンネルのイベント発生条件	
4.2.22	Do_EvtTrig	Long	R	デジタル出力 (Do) 各チャンネルのイベント発生条件	
4.2.23	Ai_EvtTrig	Long	R	アナログ入力 (Ai) 各チャンネルのイベント発生条件	
4.2.24	Ao_EvtTrig	Long	R	アナログ出力 (Ao) 各チャンネルのイベント発生条件	
4.2.25	Ai_EvtJudgeVal	Long	R	アナログ入力 (Ai) 各チャンネルのイベント発生検出差分値	
4.2.26	Ai_EvtDetecTm	Long	R	アナログ入力 (Ai) 各チャンネルのイベント発生検出インターバル(msec)	
4.2.27	Msg1	String	R	メッセージ 1	
4.2.28	Msg2	String	R	メッセージ 2	
4.2.29	KeepAliveTm	Long	R	イベントデータの発生インターバル	
4.2.30	ErrWindowStop	Long	RW	デバッグダイアログ表示フラグ	0

属性: R (参照のみ)、RW (参照／設定可)

4.2.2 ModelType プロパティ

装置の型式名称を取得します。

構文

object.ModelType

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

データ型

文字列型 (String)

解説

ModelType プロパティは、**GetMachineInfo** メソッド実行後に参照可能です。

(例)

```
TK00401.GetMachineInfo
Text1.Text = TK00401.ModelType
```

(取得例)　TK0040A

4.2.3 FirmwareVersion プロパティ

本機のファームウェアバージョンを返します。

構文

object.FirmwareVersion

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

データ型

文字列型 (String)

解説

FirmwareVersion プロパティは、GetMachineInfo メソッド実行後に参照可能です。

(例)

```
TK00401.GetMachineInfo
Text1.Text = TK00401.FirmwareVersion
```

(取得例)　v1.00

４.２.４ MachineName プロパティ

装置の機器名称を取得します。

構文

object.MachineName

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

データ型

文字列型 (String)

解説

MachineName プロパティは、**GetMachineInfo** メソッド実行後に参照可能です。

(例)

```
TK00401.GetMachineInfo
Text1.Text = TK00401.MachineName
```

(取得例)　MyCpuName

4.2.5 MacAddress プロパティ

装置のMACアドレスを返します。

構文

object.MacAddress

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

データ型

文字列型 (String)

解説

MacAddress プロパティは、**GetMachineInfo** メソッド実行後に参照可能です。

(例)

```
TK00401.GetMachineInfo
Text1.Text = TK00401.MacAddress
```

(取得例)　0004b9000000

4.2.6 GetIpAddress プロパティ

装置から取得したIPアドレスを返します。

構文

object.GetIpAddress

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

データ型

文字列型 (String)

解説

GetIpAddress プロパティは、**GetMachineInfo** メソッド実行後に参照可能です。

(例)

```
TK00401.GetMachineInfo
Text1.Text = TK00401.GetIpAddress
```

(取得例)　192.168.0.200

4.2.7 StartUpMode プロパティ

本機の起動状態(H/S)を返します。

構文

object.StartUpMode

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

データ型

文字列型 (String)

解説

StartUpMode プロパティは、**GetMachineInfo** メソッド実行後に参照可能です。

(取得値の解説)

H: 電源或はリセットスイッチ ON による起動の場合

S: リセットコマンド或はシステム異常自己診断検出自動リセットによる起動の場合

(例)

```
TK00401.GetMachineInfo
Text1.Text = TK00401.StartUpMode
```

(取得例)　H

4.2.8 KernelTime プロパティ

装置のカーネルタイムカウンタ値(秒)を返します。

構文

object.KernelTime

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

データ型

文字列型 (String)

解説

KernelTime プロパティは、**GetMachineInfo** メソッド実行後に参照可能です。

(例)

```
TK00401.GetMachineInfo
Text1.Text = TK00401.KernelTime
```

(取得例)　1234.567

4.2.9 KcxIO_Timeout プロパティ

各メソッド実行時のタイムアウト時間(ミリ秒)を設定します。値の取得も可能です。

構文

object.**KcxIO_Timeout** [= value]

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
value	Long	装置との通信タイムアウト時間(msec)を指定します。

データ型

長整数型 (Long)

解説

KcxIO_Timeout プロパティは、装置へのコマンド送信時に応答タイムアウトに使用されです。

(例)

```
Text1.Text = TK00401.KcxIO_Timeout
```

(取得例) 5000

(設定例) 3 秒に設定する場合

```
TK00401.KcxIO_Timeout = 3000
```

４.２.１０ KcxIO_IpAddress プロパティ

通信する装置のIPアドレスを設定します。値の取得も可能です。

構文

object.**KcxIO_IpAddress** [=value]

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
value	String	装置のIPアドレスを指定します。

データ型

文字列型 (String)

解説

本設定は OCX 内部で送信する装置アドレスとして使用されますので、設定したアドレスが TK0040 本体の設定アドレスと同じでないと通信できませんのでご注意下さい。

KcxIO_IpAddress プロパティは、装置へのコマンド送信時に送信先アドレスに使用されです。

(例)

```
Text1.Text = TK00401.KcxIO_IpAddress
```

(取得例) 192.168.0.200

(設定例) 192.168.0.199 に設定する場合

```
TK00401.KcxIO_IpAddress = "192.168.0.199"
```

４.２.１１ KcxIO_Port プロパティ

接続する装置のポート番号を指定します。値の取得も可能です。

構文

object.**KcxIO_Port** [= value]

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
value	Long	接続する装置のポート番号。 デフォルト値は 20000 です。

データ型

長整数型(Long)

解説

本設定はOCX内部で送信する装置のポート番号として使用されますので、設定したポート番号がTK0040本体の設定ポート番号と同じでないと通信できませんのでご注意下さい。

KcxIO_Portプロパティは、装置へのコマンド送信時に送信先ポートに使用されです。

(例)

```
TK00401.GetMachineInfo
Text1.Text = TK00401.KcxIO_Port
```

(取得例) 20000

(設定例) 20002 に設定する場合

```
TK00401.KcxIO_Port = 20002
```

4.2.12 MyPC_Port プロパティ

使用するローカルポートを設定します。値の取得も可能です。

構文

object. **MyPC_Port** [= value]

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
value	Long	データの送信に使うローカル ポートを指定します。特定のポートを必要とするアプリケーションでなければ、0 を指定(デフォルト値)してください。0 を指定すると、任意のポートが選択されます。

データ型

長整数型 (Long)

解説

MyPC_Port プロパティは、装置へのコマンド送信時に送信側ポート番号として使用されます。

(例)

```
Text1.Text = TK00401.MyPC_Port
```

(取得例)　10000

(設定例)　30000 に設定する場合

```
TK00401.MyPC_Port = 30000
```

４.２.１３ Di_Stat プロパティ

接点入力(Di)各チャンネルの状態を返します。

構文

object.**Di_Stat**(ch)

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
ch	Long	チャンネル番号(1～6)

データ型

長整数型(Long)

値	内容
正値(≧0)	正常
負値(<0)	チャンネル番号が範囲外の場合、不正パラメータエラー (-400:エラーコード一覧参照)

解説

Di_Stat プロパティは、**GetIoData** メソッド実行後に参照可能です。

(例)

```
TK00401.GetIoData
Text1.Text = TK00401.Di_Stat(1)
```

(取得例)

1　(ON)

0　(OFF)

４.２.１４ Di_TmStat プロパティ

接点入力(Di)各チャンネルの ON 保持状態を返します。

構文

object.**Di_TmStat**(ch)

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
ch	Long	チャンネル番号(1～6)

データ型

長整数型(Long)

値	内容
正値(≧0)	正常
負値(<0)	チャンネル番号が範囲外の場合、不正パラメータエラー (-400:エラーコード一覧参照)

解説

Di_TmStat プロパティは、**GetIoData** メソッド実行後に参照可能です。

(例)

```
TK00401.GetIoData
Text1.Text = TK00401.Di_TmStat(1)
```

(取得例)

1　(ON)

0　(OFF)

4.2.15 Di_TmVal プロパティ

接点入力(Di)各チャンネルのON保持状態(カウントダウン値)を返します。

構文

object.**Di_TmVal**(ch)

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
ch	Long	チャンネル番号(1～6)

データ型

長整数型(Long)

値	内容
正値(≧0)	正常
負値(<0)	チャンネル番号が範囲外の場合、不正パラメータエラー (-400:エラーコード一覧参照)

解説

Di_TmVal プロパティは、**GetDiHoldTm** メソッド実行後に参照可能です。

(例)

```
TK00401.GetDiHoldTm
Text1.Text = TK00401.Di_TmVal(1)
```

(取得例) 12

４.２.１６ Di_Count プロパティ

接点入力(Di)各チャンネルの開閉回数を返します。

構文

object.**Di_Count**(ch)

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
ch	Long	チャンネル番号(1～6)

データ型

長整数型(Long)

値	内容
正値(≧0)	正常
負値(<0)	チャンネル番号が範囲外の場合、不正パラメータエラー (-400:エラーコード一覧参照)

解説

Di_Count プロパティは、**GetIoData** メソッド実行後に参照可能です。

(例)

```
TK00401.GetIoData
Text1.Text = TK00401.Di_Count(1)
```

(取得例)　77

4.2.17 Do_Stat プロパティ

リレー出力(Do)各チャンネルの状態を返します。

構文

object.**Do_Stat**(ch)

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
ch	Long	チャンネル番号(1～4)

データ型

長整数型(Long)

値	内容
正値(≧0)	正常
負値(<0)	チャンネル番号が範囲外の場合、不正パラメータエラー (-400:エラーコード一覧参照)

解説

Do_Stat プロパティは、**GetIoData** メソッド実行後に参照可能です。

(例)

```
TK00401.GetIoData
Text1.Text = TK00401.Do_Stat(1)
```

(取得例)

1　(ON)

0　(OFF)

4.2.18 Ai_Val プロパティ

アナログ入力各チャンネルの状態を返します。

構文

object.**Ai_Val**(ch)

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
ch	Long	チャンネル番号 (1～4)

データ型

長整数型 (Long)

値	内容
正値 (≧0)	正常
負値 (<0)	チャンネル番号が範囲外の場合、不正パラメータエラー (-400:エラーコード一覧参照)

解説

Ai_Val プロパティは、**GetIoData** メソッド実行後に参照可能です。

(例)

```
TK00401.GetIoData
Text1.Text = TK00401.Ai_Val(1)
```

(取得例) 1023

４.２.１９ Ao_Val プロパティ

アナログ出力各チャンネルの状態を返します。

構文

object.**Ao_Val**(ch)

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
ch	Long	チャンネル番号 (1～2)

データ型

長整数型 (Long)

値	内容
正値 (≧0)	正常
負値 (<0)	チャンネル番号が範囲外の場合、不正パラメータエラー (-400:エラーコード一覧参照)

解説

Ao_Val プロパティは、**GetIoData** メソッド実行後に参照可能です。

(例)

```
TK00401.GetIoData
Text1.Text = TK00401.Ao_Val(1)
```

(取得例) 255

４.２.２０ PWMo_Val プロパティ

PWM 出力各チャンネルのスケール値を返します。

構文

object.**PWMo_Val**(ch)

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
ch	Long	チャンネル番号 (1～3)

データ型

長整数型 (Long)

値	内容
正値 (≧0)	正常
負値 (<0)	チャンネル番号が範囲外の場合、不正パラメータエラー (-400:エラーコード一覧参照)

解説

PWMo_Val プロパティは、**GetIoData** メソッド実行後に参照可能です。

(例)

```
TK00401.GetIoData
Text1.Text = TK00401.PWMo_Val(1)
```

(取得例) 4567

4.2.21 Di_EvtTrig プロパティ

接点入力各チャンネルのイベント発生条件を返します。

構文

object.**Di_EvtTrig**(ch)

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
ch	Long	チャンネル番号 (1～6)

データ型

長整数型 (Long)

値	内容
正値 (≧0)	正常
負値 (<0)	チャンネル番号が範囲外の場合、不正パラメータエラー (-400:エラーコード一覧参照)

解説

Di_EvtTrig プロパティは、**GetDioEventTrig** メソッド実行後に参照可能です。

(例)

```
TK00401.GetDioEventTrig
Text1.Text = TK00401.Di_EvtTrig(1)
```

(取得例)

0 (機能無効)

1 (on)

2 (off)

3 (on/off)

4.2.22 Do_EvtTrig プロパティ

デジタル(リレーorTR)出力各チャンネルのイベント発生条件を返します。

構文

object.**Do_EvtTrig**(ch)

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
ch	Long	チャンネル番号(1～4)

データ型

長整数型(Long)

値	内容
正値(≧0)	正常
負値(<0)	チャンネル番号が範囲外の場合、不正パラメータエラー (-400:エラーコード一覧参照)

解説

Do_EvtTrig プロパティは、**GetDioEventTrig** メソッド実行後に参照可能です。

(例)

```
TK00401.GetDioEventTrig
Text1.Text = TK00401.Do_EvtTrig(1)
```

(取得例)

0 (機能無効)

1 (on)

2 (off)

3 (on/off)

4.2.23 Ai_EvtTrig プロパティ

アナログ入力各チャンネルのイベント発生条件を返します。

構文

object.**Ai_EvtTrig**(ch)

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
ch	Long	チャンネル番号 (1～4)

データ型

長整数型 (Long)

値	内容
正値 (≧0)	正常
負値 (<0)	チャンネル番号が範囲外の場合、不正パラメータエラー (-400:エラーコード一覧参照)

解説

Ai_EvtTrig プロパティは、**GetAioEventTrig** メソッド実行後に参照可能です。

(例)

```
TK00401.GetAioEventTrig
Text1.Text = TK00401.Ai_EvtTrig(1)
```

(取得例)

0 (機能無効)

1 (有効)

4.2.24 Ao_EvtTrig プロパティ

アナログ出力各チャンネルのイベント発生条件を返します。

構文

object.**Ao_EvtTrig**(ch)

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
ch	Long	チャンネル番号 (1～2)

データ型

長整数型 (Long)

値	内容
正値 (≧0)	正常
負値 (<0)	チャンネル番号が範囲外の場合、不正パラメータエラー (-400:エラーコード一覧参照)

解説

Ao_EvtTrig プロパティは、**GetAioEventTrig** メソッド実行後に参照可能です。

(例)

```
TK00401.GetAioEventTrig
Text1.Text = TK00401.Ao_EvtTrig(1)
```

(取得例)

0 (機能無効)

1 (有効)

４.２.２５ Ai_EvtJudgeVal プロパティ

アナログ出力のｲﾍﾞﾝﾄ発生検出差分値を返します。(チャンネル共通)

構文

object.Ai_EvtJudgeVal

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

データ型

長整数型(Long)

解説

Ai_EvtJudgeVal プロパティは、**GetAioEventTrig** メソッド実行後に参照可能です。

(例)

```
TK00401.GetAioEventTrig
Text1.Text = TK00401.Ai_EvtJudgeVal
```

(取得例) 100

４.２.２６ Ai_EvtDetecTm プロパティ

アナログ出力のイベント発生検出インターバル(msec)を返します。(チャンネル共通)

構文

object.Ai_EvtDetecTm

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

データ型

長整数型(Long)

解説

Ai_EvtDetecTm プロパティは、**GetAioEventTrig** メソッド実行後に参照可能です。

(例)

```
TK00401.GetAioEventTrig
Text1.Text = TK00401.Ai_EvtDetecTm
```

(取得例) 180

４.２.２７ Msg1 プロパティ

Msg1 に設定された文字列を返します。

構文

object.**Msg1**

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

データ型

文字列型 (String)

解説

Msg1 プロパティは、**GetMsg1** メソッド実行後に参照可能です。

(例)

```
TK00401.GetMsg1
Text1.Text = TK00401.Msg1
```

(取得例)　abcd-1234

４.２.２８ Msg2 プロパティ

Msg2 に設定された文字列を返します。

構文

object.**Msg2**

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

データ型

文字列型 (String)

解説

Msg2 プロパティは、**GetMsg2** メソッド実行後に参照可能です。

(例)

```
TK00401.GetMsg2
Text1.Text = TK00401.Msg2
```

(取得例)　efgh-6789

４.２.２９ KeepAliveTm プロパティ

イベントデータ発生インターバル時間(秒)を返します。

構文

object.**KeepAliveTm**

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。

データ型

長整数型 (Long)

解説

KeepAliveTm プロパティは、**Get** KeepAliveTm メソッド実行後に参照可能です。

(例)

```
TK00401.GetKeepAliveTm
Text1.Text = TK00401.KeepAliveTm
```

(取得例)　600

４.２.３０ ErrWindowStop プロパティ

プログラム実行時に、エラーが発生したときにデバッグダイアログ画面を表示するフラグです。

構文

object. **ErrWindowStop** [= value]

構文の指定項目は次のようになります。

指定項目	データ型	内　容
object		オブジェクトを指すオブジェクト式です。
value	Long	1 を指定すると、デバッグダイアログ画面を表示するモードに設定されます。 デフォルトは 0 (無効) です。0 に設定されている場合は、メソッドの戻り値を使用して、エラーの有無を判定することができます。

データ型

長整数型 (Long)

解説

(取得例)　0

(設定例)　TK00401.ErrWindowStop = 1

ErrWindowStop プロパティは、プログラム実行時のデバッグの方法を選択するフラグとして機能します。デフォルト値は 0 (無効) になっており、この場合、メソッド実行時にエラーが発生した場合には、メソッドの戻り値にエラーコード (負値) が返ります。以下にコーディング例を示します。

```
Private Sub func1()
   Dim status As Long
   status = TK00401.GetIoData        ' メソッドの実行
   If ( status > 0 ) Then
      Text1.Text = TK00401.Di_Stat(0)    ' 1ch 目のデジタル入力状態表示
      Text2.Text = TK00401.Ai_Val(0)     ' 1ch 目のアナログ入力値表示
   Else
      Text3.Text = status         'エラーコードを表示
   End If
End Sub
```

本プロパティを１(有効)に設定すると、メソッド実行時にエラーが発生した場合には以下のようにエラーダイアログにエラーコードとエラーメッセージが表示され処理が中断されます。

エラーダイアログで、終了ボタンを押すと、プログラムが終了します。また、デバッグボタンを押すとVBのデバッグ機能でエラーの発生したコード行が表示されます。

(参考)
本プロパティを有効にしている場合には、On Error GoTo ラベル構文を使用することでエラーダイアログの表示を抑止しながらエラー処理を行うことが可能です。以下にコーディング例を示します。

```
Private Sub func1()
   Dim msg$
   On Error GoTo ErrNetCom   ' エラー発生時にラベル(ErrNetCom)へジャンプ
  TK00401.GetIoData        ' メソッドの実行
  Text1.Text = TK00401.Di_Stat(0)     ' 1ch 目のデジタル入力状態表示
  Text2.Text = TK00401.Ai_Val(0)     ' 1ch 目のアナログ入力値表示
  Text3.Text = TK00401.Ai_Val(1)     ' 2ch 目のアナログ入力値表示
ErrNetCom:
   msg$ = "エラー:" & _
            """ & Err.Description & "" " & _
            "(" & Format(Err.Number) & ")"
   sbrMain.Panels(1).Text = msg$       'ステータスバーにエラーを表示する
End Sub
```

４.３ エラーコード一覧

コード*	エラー内容
-100	TK0040 との通信タイムアウトが発生しました。
-200	ソケット作成時にエラーが発生しました。
-201	ソケットバインド時にエラーが発生しました。
-202	ソケットクローズ時にエラーが発生しました。
-203	送信 SELECT でエラーが発生しました。
-204	受信 RECVFROM でエラーが発生しました。
-300	パケットID取得時にエラーが発生しました。
-301	パケットコマンド取得時にエラーが発生しました。
-302	パケット不正データ取得エラーが発生しました。
-400	不正なパラメータが指定されています。

*メソッドの返り値で取得する場合には負号(-)が付きます。

付録. サンプルプログラムの使用法

Visual Basicサンプルプログラムファイル(KCX_VB_TK0040_v???.lzh)を弊社サイトからダウンロードします。(???はバージョン番号を示します)
ダウンロードしたファイルを解凍すると本OCXを使用したVBサンプルインストーラとソースコードが以下のファイル名で同梱されています。

KCX_VB_TK0040_v???.EXE (VBサンプルインストーラ)
(???はバージョン番号を示します)
TK0040_Vbsample01src (VBサンプルソース)

VB サンプルインストーラを実行して指示に従ってインストールして下さい。
インストールフォルダに、TK0040_VBsample??.EXE (VB サンプル実行ファイル) が作成されます。
ここでは、サンプルプログラムの使用法を説明します。(??はバージョン番号を示します)

１．環境設定ダイアログの設定

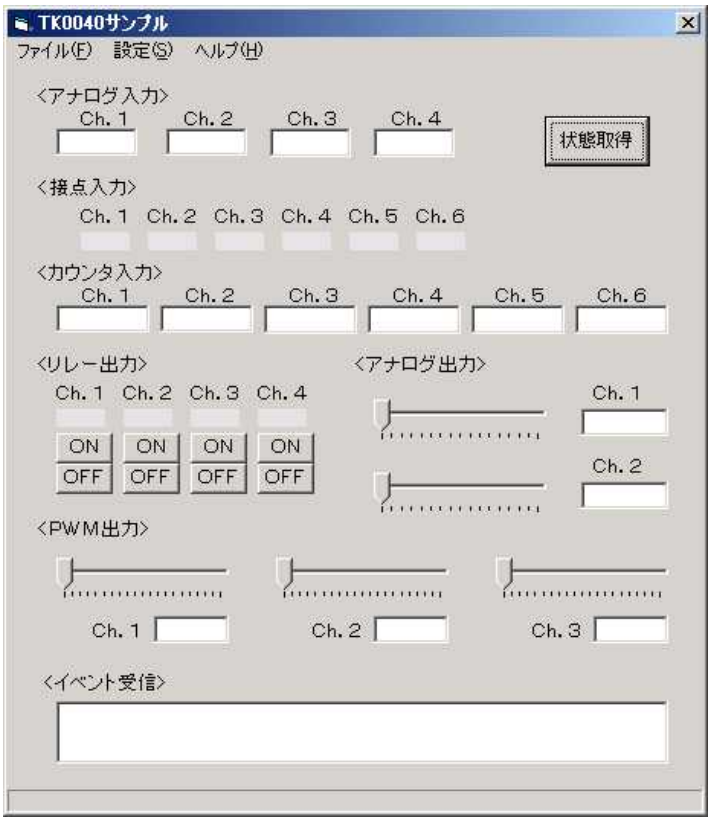
インストールしたフォルダのサンプルプログラムをクリックして起動すると、以下の環境設定ダイアログが表示されます。

各入力項目は以下のようになります。
リモートホスト: 通信する TK0040 装置に設定されている IP アドレスです。
リモートポート: 通信する TK0040 装置に設定されているコントロールポート番号です。
データ取得間隔: タイマーコントロールで定期的に TK0040 装置からデータ取得するインターバル時間 (秒) です。
タイムアウト: TK0040 装置に通信コマンドを送信時の応答タイムアウト時間 (ミリ秒) です。
イベントポート: TK0040 装置から送信されるイベントパケットを受信するＰＣ側ポート番号です。

上記のうち、リモートホスト、リモートポート、イベントポートは、TK0040 の環境設定と同じ値でないと通信することが出来ませんので、よく確認して設定して下さい。
環境設定値が正しく設定出来たら、「ＯＫ」ボタンをクリックして下さい。

２．メイン情報表示画面の使い方

環境設定ダイアログで正しく接続情報を入力すると、以下のようにメイン情報表示画面が表示されます。



各項目の説明を以下に示します。

＜アナログ入力＞：　TK0040 装置から取得したアナログ入力値をチャンネル別に表示します。

＜接点入力＞：　TK0040 装置から取得した接点入力状態をチャンネル別に表示します。

＜カウンタ入力＞：　TK0040 装置から取得したカウンタ入力値をチャンネル別に表示します。

＜リレー出力＞：　TK0040 装置のリレー出力（デジタル出力）をＯＮ／ＯＦＦ操作します。また、TK0040 装置から取得したデジタル出力の現在値をチャンネル別に表示します。

＜アナログ出力＞：　スライドバーで、TK0040 装置のアナログ出力に値を設定します。また、TK0040 装置から取得したアナログ出力の現在値をチャンネル別に表示します。

＜PWM出力＞：　スライドバーで、TK0040 装置のPWM出力に値を設定します。また、TK0040 装置から取得したPWM出力の現在値をチャンネル別に表示します。

＜イベント受信＞：　TK0040 装置から取得したイベントパケットをそのまま表示します。

状態取得ボタン：　TK0040 装置にデータ取得コマンド、操作コマンドを送信して、応答ステータス

を上記の各項目に表示します。

次に、メニューバーの各項目の説明を以下に示します。

[ファイル]メニュー：

終了(X)：本サンプルプログラムを終了します。

[設定]メニュー：

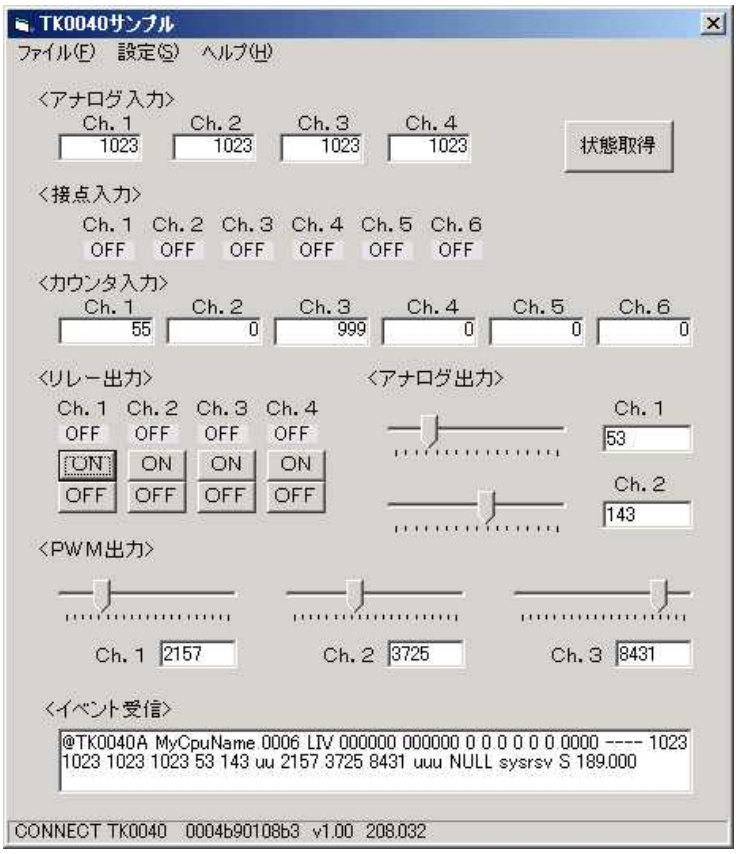
環境(F)：環境設定ダイアログを表示します。接続する装置アドレス等を変更する場合に使用します。

タイマー(T)：タイマーコントロールを有効にします。定期的にデータ取得を行う場合は選択してチェックを入れます。

[ヘルプ]メニュー：

バージョン情報(A)：本サンプルプログラムのバージョン情報を表示します。

TK0040 装置と通信して現在の入力状態値を取得するには、「状態取得」ボタンをクリックして下さい。正常に通信が成功すれば以下の画面のように取得状態値が表示されます。



株式会社エスアイ創房　〒150-0036 東京都渋谷区南平台町 2-12 久保ビル 703　　http://www.karacrix.jp