Roboticsware



Automation シリーズ

IPLink インターフェース

～詳細マニュアル～

RWMN-5-21-953

ようこそ **4**
マニュアルの読み方 4
対応インターフェース一覧 5
仮想デバイスについて 6
1章　IPLINK インターフェース仕様 **7**
1－1　IPLINK 接続の概要 7
IPLink とは 7
1－2　IPLINK サーバとしての利用 8
IPLink インターフェースの有効化 8
1－3　IPLINK クライアントとしての利用 9
FA シリーズからの IPLink クライアント接続 9
Visual Basic からの IPLink クライアント接続 9
開発環境について 10
コンパイルについて 11
1－4　IPLINK の利用手順 12
IPLink を利用する際の基本手順 12
2章　はじめての IPLINK 接続 **13**
2－1　はじめての接続（VISUAL BASIC） 13
概要 13
ステップ1（サーバ設定） 13
ステップ2（クライアント設定） 16
2－2　リモート接続 24
概要 24
クライアントのセットアップ 25
接続確認 26
3章　機能解説 **27**
インターフェース一覧 27
一般的なプログラムの処理フロー 28
クライアントキャッシュとサーバキャッシュ 29
4章　VISUAL BASIC による IPLINK 接続の実践 **31**
概要 31
4－1　モニタリング（SAMPLE01） 33
概要 33
プログラムの要点 34
通信切断の検証 36
4－2　データの読み込み（SAMPLE02） 38
概要 38
プログラムの要点 39
動作確認 41
4－3　データの書き込み（SAMPLE03） 42
概要 42
プログラムの要点 42
動作確認 43
4－4　大量なデータの扱い方（SAMPLE04） 44

概要 44
プログラムの要点 45
４－５　複数フォームでの扱い方（SAMPLE05） 46
概要 46
プログラムの要点 47
動作確認 47
４－６　二重化運用サーバへの接続（SAMPLE06） 48
概要 48
サーバ設定 49
プログラム 51
動作確認 57
５章　リファレンス 58
５－１　機能一覧 58
５－２　メソッド・プロパティ 59
NetworkParam プロパティ 59
Online プロパティ 59
Update (.Net の場合：CtlUpdate) プロパティ 60
AddTag メソッド 60
AddTags メソッド 61
ActiveTag メソッド 61
ActiveTags メソッド 62
DelTag メソッド 62
DelTags メソッド 63
ReadVal メソッド 63
ReadVals メソッド 64
WriteVal メソッド 65
WriteVals メソッド 66
AccessPath プロパティ 67
State (.Net の場合：CtlState) プロパティ 68
ErrMessage プロパティ 68
５－３　イベント 69
ValueStateChanged イベント 69
ValueChanged イベント 70
ValueBitOn/ValueBitOff イベント 71
StateChanged イベント 72

ようこそ

マニュアルの読み方

本書「IPLink インターフェース詳細マニュアル」は、対応インターフェースの一つである「IPLink インターフェース」を用いて、IPLink を介して他のアプリケーションと接続する方法について記述されたマニュアルです。IPLink インターフェースの利用により、IPLink に対応しているアプリケーション同士での接続や、Visual Basic で開発した独自アプリケーションなどから本パッケージに接続することが可能です。このマニュアルでは、特に Visual Basic からの IPLink 接続に関する詳細な解説が記述されています。

対応インターフェース一覧

FA シリーズでは、パッケージの種類によって対応可能なインターフェースが異なります。お使いのパッケージが IPLink に対応しているか否かは、下表を参照してご確認ください。

■インターフェース対応表

インターフェース	Panel	Studio+	Studio	Server	Logger	Driver	Report	Trend	Engine
OPC	○	○	○	○					
DDE	○	○	○	○					
SuiteLink		○	○	○					
IPLink	○	○	○	○	○	○	○	○	
Panel	○								

仮想デバイスについて

現在、本パッケージは 100 機種相当との通信に対応しており、その対応機種の 1 つとして、「仮想デバイス」があります。「仮想デバイス」は PLC をシミュレートしたアドレス空間であり、特定の PLC との通信を行わないという点を除き、標準的な PLC デバイスと同等に扱えます。本マニュアルでは、PLC との通信を例に解説する場面で特定の PLC に限定して説明することを避けるため、「仮想デバイス」を用いて説明を進めます。「仮想デバイス」での扱いを理解すれば、その他の標準的な PLC でも同様の手順で設定が行えます。

仮想デバイス一覧

デバイス	アドレス	説明	サイズ	読み	書き
入力リレー	X0000～X9999	10 進数	ビット	○	○
出力リレー	Y0000～Y9999	10 進数	ビット	○	○
内部リレー	M0000～M9999	10 進数	ビット	○	○
特殊リレー	SM0000～SM9999 **(※1)**	10 進数	ビット	○	○
データレジスタ	D0000～D9999	10 進数	ワード	○	○
特殊レジスタ	SD0000～SD9999 **(※2)**	10 進数	ワード	○	○

(※1)特殊リレー（SM）について

SM0000	10ms ごとに On/Off	SM0004	1000ms ごとに On/Off
SM0001	50ms ごとに On/Off。	SM0005	2000ms ごとに On/Off
SM0002	100ms ごとに On/Off	SM0006	3000ms ごとに On/Off
SM0003	500ms ごとに On/Off	SM0007	5000ms ごとに On/Off

(※2)特殊レジスタ（SD）について

SM0000	10ms ごとに On/Off	SD0004	1000ms ごとにインクリメント
SD0001	50ms ごとにインクリメント	SD0005	2000ms ごとにインクリメント
SD0002	100ms ごとにインクリメント	SD0006	3000ms ごとにインクリメント
SD0003	500ms ごとにインクリメント	SD0007	5000ms ごとにインクリメント

SD0008～SD0015 は、それぞれ SD0000～SD0007 の値を用い、0～100 の間でサインカーブを描く。

1章　IPLink インターフェース仕様

1－1　IPLink 接続の概要

IPLink とは

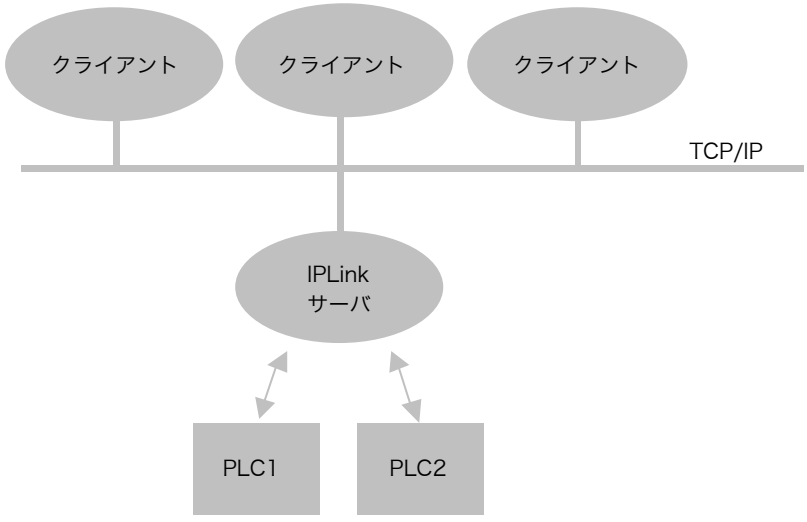
IPLink とは、本パッケージ独自の TCP/IP ベースネットワークプロトコルで、クライアント／サーバ型の通信方式です。本パッケージは、IPLink サーバ、IPLink クライアント両方の機能を備えており、アプリケーション同士を通信接続させることができます。また、Visual Basic などで開発した独自のクライアントアプリケーションにおいては、IPLink クライアントライブラリの利用によって、IPLink サーバに接続することができます。

IPLink の特徴は以下のとおりです。

●クライアントはネットワーク上のどの PC（Windows パソコン）からでも接続可能
IPLink クライアントからの接続では、IP アドレスとポート番号によって IPLink サーバを特定します。IPLink サーバが動作している同一の PC 上はもちろん、リモートの PC など、同一の TCP/IP ネットワークに接続されたノードであればどこからでも接続が可能です。例えば、社内 LAN や構内 LAN 経由、あるいはインターネット経由などで PLC デバイスを監視することができます。

●複数クライアントからの同時接続が可能
IPLink サーバに対する接続は、複数のクライアントから同時に行うことができます。たとえば、IPLink クライアントとして動作する監視画面プログラムを Visual Basic で 1 つ作成し、そのプログラムを複数の PC 上で同時に実行した場合、各 PC の画面では同等の設備監視が可能となります。

●異機種／複数台 PLC との同時接続が可能
クライアントからはタグ単位で接続を行います。各 PLC デバイスの値を内部的にタグとして置き換えることにより、複数メーカー、異機種が混在するような複数台 PLC 構成においても、クライアント側から見ると PLC が 1 台のみの場合と同等に扱うことができます。



1－2　IPLink サーバとしての利用

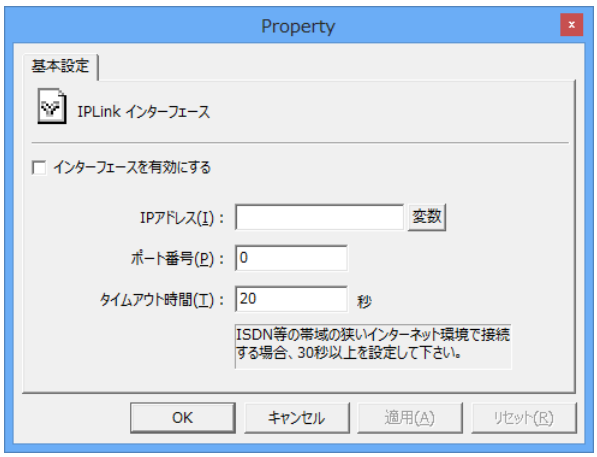
IPLink インターフェースの有効化

アプリケーションを IPLink サーバとして動作させる場合には、IPLink インターフェースを有効にする必要があります。インターフェースの有効化は、以下の手順によって行います。

1．IPLink のプロパティ画面を開く
画面左側のツリーから「Interface」アイコンを開き、「IPLink」を右クリックして「プロパティ」を選択します。



2．プロパティ設定
プロパティ画面から、IPLink サーバの IP アドレス 及び ポート番号を指定して、「インターフェースを有効にする」にチェックを入れてください。



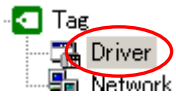
尚、複数のネットワークカードが存在する時など、PC に複数の IP アドレスがある場合を除き、IP アドレスを明記する必要はありません。指定が無い場合には自動的にデフォルト IP アドレスが設定されます。又、ポート番号にはシステム構成上使用可能な任意の番号を指定してください。

1－3　IPLink クライアントとしての利用

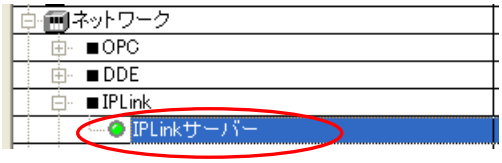
FA シリーズからの IPLink クライアント接続

本パッケージは、IPLink サーバとしての機能の他に、IPLink クライアントとしても動作させることができます。IPLink サーバとの接続は、ユニットに「IPLink サーバ」を設定する事によって行います。

●IPLink サーバ接続用のユニット追加
画面左のツリーから Driver を右クリックしてメニューを表示し、「ユニット追加」を選択します。



ドライバー覧ダイアログから、ネットワーク／IPLink／IPLink サーバを選択します。



ユニットの追加が完了したら、接続先 IPLink サーバの情報を（IP アドレス、ポート番号）、ユニットの通信設定ダイアログから入力してください。

Visual Basic からの IPLink クライアント接続

Visual Basic 等の開発言語によって、独自の IPLink クライアントアプリケーションを開発することができます。

クライアントからの接続には、IPLink サーバへ接続するためのライブラリ「ClientIPLink5.ocx」を利用します。ライブラリは「ClientIPLink5.ocx」のみで動作し、ランタイムの配布はフリーです。ライブラリのセットアップに関する詳細は「インストールガイド」を参照下さい。

以降、本書では、Visual Basic による IPLink 接続に重点を置いて解説します。



旧バージョンである「ClientIPLink4.ocx」を使用して IPLink クライアントアプリケーションを作成している場合、Ver5 の IPLink サーバーへの接続は保証されていません。そのため、移行が必要になります。

移行方法は、「ClientIPLink4.ocx」を「ClientIPLink5.ocx」へ張り替えるのみで、プログラムなどを変更する必要はありません。

開発環境について

IPLink クライアントアプリケーションを開発する際に、使用できる開発環境で現在サポートしているのは以下の環境となります。

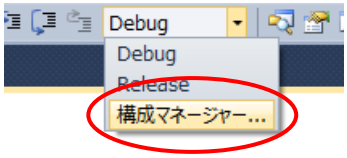
・VisualBasic 6.0
・VisualBasic .net（VisualStudio2003/2005/2008/2010）

また、FA-Engine の OCX は 32Bit（x86）構成になっています。そのためアプリケーションのコンパイルを行う場合、32Bit（x86）指定でコンパイルを行う必要があります。コンパイルの方法については「コンパイルについて」の項目を参照してください。

コンパイルについて

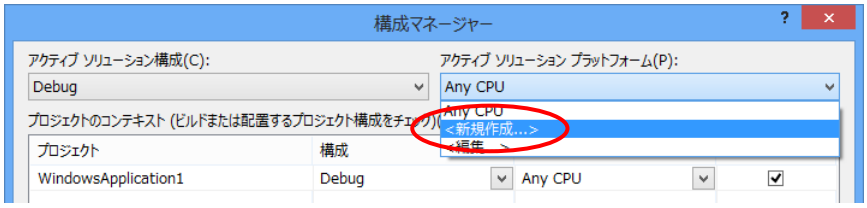
IPLink の OCX は 32Bit（x86）構成になっています。そのためクライアントアプリケーションを作成する際に、コンパイルを行う場合、32Bit（x86）指定でコンパイルを行う必要があります。以下の手順で構成を変更できます。

1. 構成マネージャーを選択する

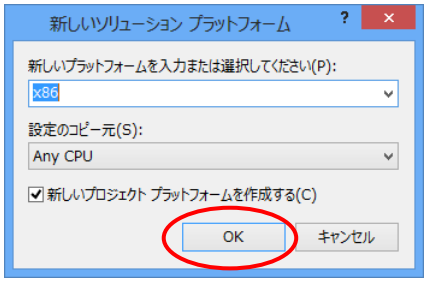


構成マネージャーが表示されない場合、[ツール] - [オプション] - [プロジェクトおよびソリューション] - [全般]を開き、[ビルド構成の詳細を表示]にチェックを入れてください。

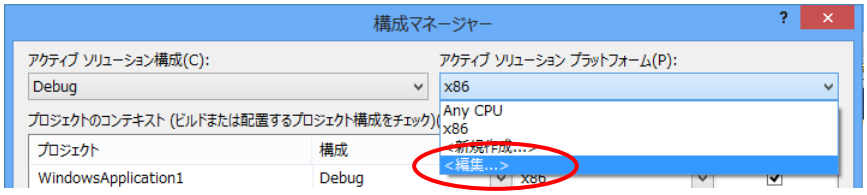
2. 下記の設定箇所から新規作成を選択する



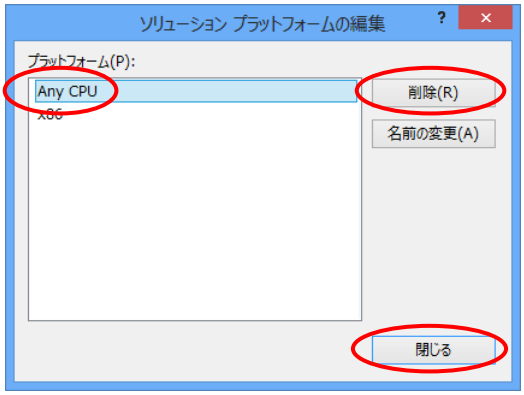
3. X86 を選択し OK を押下する



4. 編集を選択する



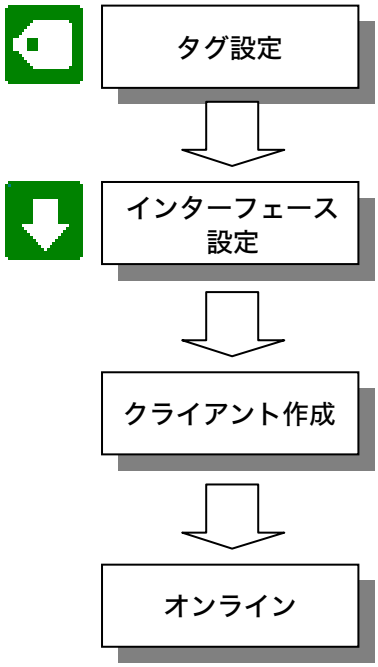
5. 「Any CPU」を選択して削除ボタンを押して設定を削除し、ダイアログを閉じる



1－4　IPLink の利用手順

IPLink を利用する際の基本手順

IPLink サーバを利用する際の基本的な手順は以下のとおりです。



PLC のどのエリアと通信を行うのかを定義します。

IPLink サーバとしてのインターフェース設定を行います。

Visual Basic などで IPLink クライアントアプリケーションを作成します。

アプリケーションをオンライン状態すると PLC との通信が始まります。
オンライン後、クライアントアプリケーションを実行します。

2章　はじめての IPLink 接続

2－1　はじめての接続（Visual Basic）

概要

この章では、Visual Basic アプリケーションから IPLink 接続を行うための入門的な手順について解説します。接続の手順は大きくわけて以下の２ステップです。

ステップ１	サーバ設定	タグの登録を行い、IPLink サーバとしての設定を行います。
ステップ２	クライアント設定	Visual Basic を使って、IPLink 経由でパッケージに接続を行い、モニタリング（タグの読み書き）を行うためのプログラムを作成します。

ステップ１（サーバ設定）



タグ設定

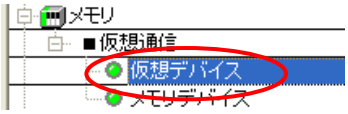
下記の手順に従ってタグの設定を行って下さい。ここでは、PLC の代わりに仮想デバイスを使用します。もし、PLC が手元にあれば、PLC のタグを設定しても構いません。

１．ユニットの追加

画面左側のツリーから「Driver」を右クリックしてメニューを表示し、「ユニット追加」を選択してください。



選択ダイアログが表示されるので、「メモリ」の「仮想通信」「仮想デバイス」を選択して、「OK」を押してください。



２．フォルダ追加

「U01」を右クリックしてメニューを表示し、「フォルダ追加」を選択します。



3．タグ追加

「F01」を右クリックしてメニューを表示し、「タグ追加」を選択します。



「T01」というタグが追加されます。

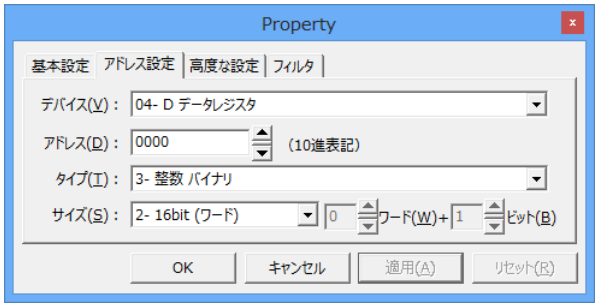
4．タグのプロパティ設定

リストから「T01」を右クリックしてメニューを表示し、「プロパティ」を選択します。



「アドレス設定」タブから「04 - D データレジスタ」を選択し、「OK」を押してください。

以下は例として、D0000（ワードレジスタ）を数値タグとして設定しています。



インターフェース設定

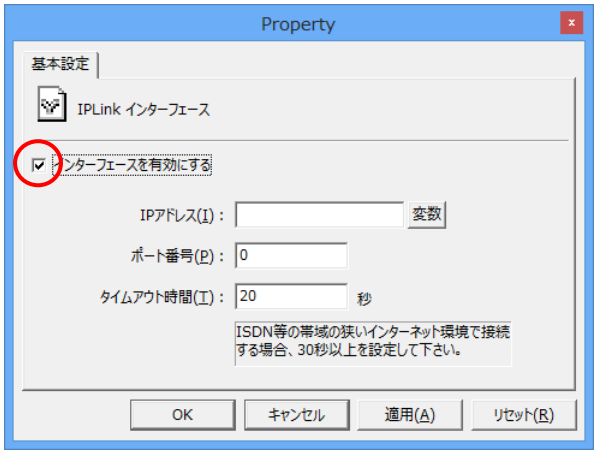
続いて、IPLink インターフェースの設定を行います。

1．左側のツリーの「IPLink」を右クリックし「プロパティ」を選択



2．プロパティ設定

IPLink サーバの IP アドレス及びポート番号を指定します。「インターフェースを有効にする」にチェックが入っていることを確認してください。



複数のネットワークカードが存在する場合など、PC に複数の IP アドレスがある場合を除き、IP アドレスを明記する必要は特にありません。アドレスの指定が無い場合、自動的にデフォルト IP アドレスが設定されます。また、ポート番号はシステム構成上使用が可能な任意の番号を指定してください。ここでは 3000 を指定します。

以上でサーバ側の設定は完了です。ここまでの作業で設定ファイルを保存し、アプリケーションをオンラインにしてください（黄色矢印）。

注意

IPLink サーバを同じ PC 内で複数起動させる場合には、各サーバがそれぞれ異なるポート番号となるように設定してください。

ヒント



オンラインには３つの状態があります。

赤：オンライン（デバック）　仮想的に通信します。PLC が手元に無いときに使用します。

黄：オンライン（モニタ）　　アプリケーションで現在値の確認や設定が行えます。

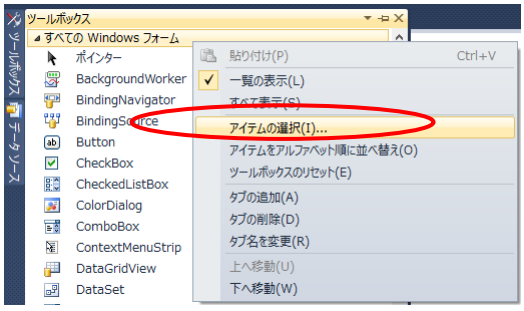
緑：オンライン（最速）　アプリケーションは消え、タスクバーへアイコンとして表示されます。

ステップ２（クライアント設定）

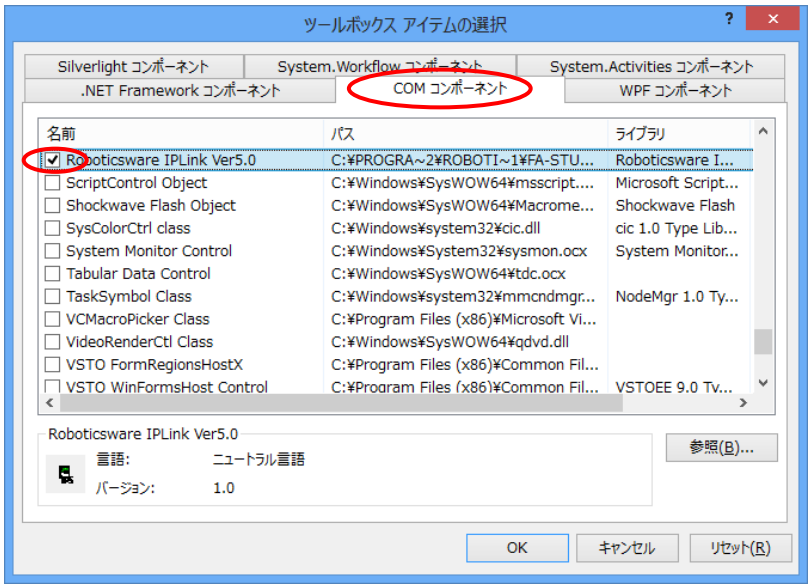
続いて、Visual Basic でクライアントアプリケーションの作成を行います。ここでは VB.Net2010 を例としています。作業される場合は実際に使用される VB のバージョンに合わせた設定を行ってください。

1．Visual Basic を起動する

2．ツールボックスで右クリックし、メニュー［アイテムの選択］を選択する

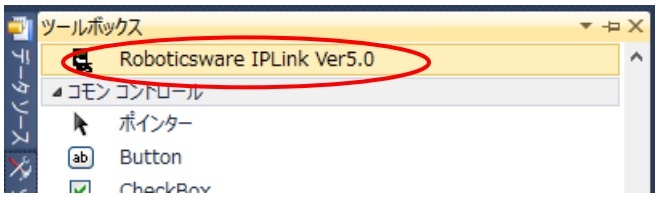


3．「Roboticsware IPLink Ver5.0」にチェックを付けて、「OK」をクリックする



4．コントロールをフォームに貼り付ける

ツールバーから「IPLink」オブジェクトを選択し、フォームに貼り付けます。

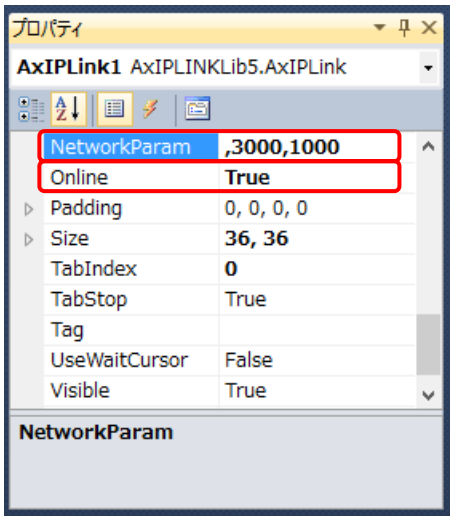


5．プロパティ設定

IPLink コントロールのプロパティに、下記の設定をおこないます。

NetworkParam : 「,3000,1000」
Online : 「True」

NetworkParam は「サーバ IP アドレス,サーバポート番号,更新周期」の順にパラメータを設定します。
Online を TRUE にすると、サーバと接続され通信可能な状態になります。

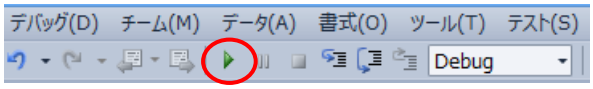


NetworkParam にて「サーバ IP アドレス,サーバポート番号,更新周期」に続けて、「通信タイムアウト（秒）,接続時タイムアウト（秒）」を設定できます。
例）「,3000,1000,10,10」

接続の確認

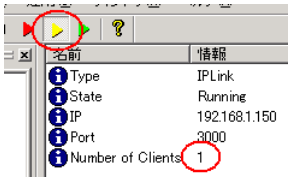
ここまでの設定で、IPLink による通信が正しく行われることを確認してみます。
Visual Basic の開始ボタンを押して、プログラムを実行して下さい。「Number of Clients」には、現在接続されているクライアントの数が表示されます。この値が「1」になっている事で、接続が正しく行われていることが確認できます。

●Visual Basic



実行ボタン

●サーバ側



クライアント接続数を確認する

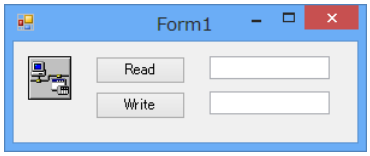
※Visual Basic を開始する前にアプリケーションがオンライン（黄色ボタン）になっていることを確認して下さい。

ReadVal/WriteVal によるワンショット通信

続いて、「ReadVal」「WriteVal」メソッドを呼び出して、ボタンクリックによるタグの読み書きを行います。

1．コントロールの貼り付け

フォームに２つのボタンと２つのテキストボックスを貼り付けます。



このボタンとテキストを、タグの読み書きに使用します。

2．コード記述

以下のコードを記述してください。

●VB.Net

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    AxIPLink1.AddTag("U01.F01.T01", True)  'タグ登録
End Sub

'読み込みボタン
Private Sub Button1_Click(ByVal sender As System.Object,
                          ByVal e As System.EventArgs) Handles Button1.Click
    Dim a As Object = Nothing
    If Not AxIPLink1.ReadVal("U01.F01.T01", a, 1) Then
        TextBox1.Text = "Error!"
    Else
        TextBox1.Text = a
    End If
End Sub

'書き込みボタン
Private Sub Button2_Click(ByVal sender As System.Object,
                          ByVal e As System.EventArgs) Handles Button2.Click
    If Not AxIPLink1.WriteVal("U01.F01.T01", TextBox2.Text) Then
        TextBox2.Text = "Error!"
    End If
End Sub
```

●VB6

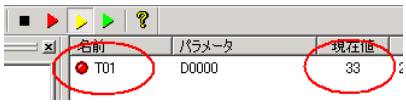
```
Private Sub Form_Load()
    IPLink1.AddTag "U01.F01.T01", True    'タグ登録
End Sub

'読み込みボタン
Private Sub Command1_Click()
    Dim a As Variant
    If Not IPLink1.ReadVal("U01.F01.T01", a, 1) Then
        Text1.Text = "Error!"
    Else
        Text1.Text = a
    End If
End Sub

'書き込みボタン
Private Sub Command2_Click()
    If Not IPLink1.WriteVal("U01.F01.T01", Text2.Text) Then
        Text2.Text = "Error!"
    End If
End Sub
```

3．プログラムの実行

Visual Basic の開始ボタンを押して、プログラムを開始して下さい。T01 タグのランプが下記のように赤色に変わっていればタグへの接続が完了しています。タグの現在値は「現在値」の欄で確認できます。



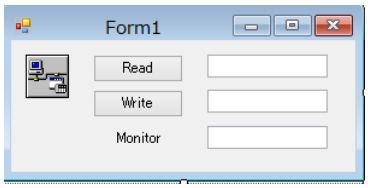
T01 タグの値を更新して Visual Basic 側の読み込みボタンを押し、値が取得できる事を確認してください。また、Visual Basic から値を更新して書き込みボタンを押し、T01 タグに値が渡されることを確認してください。

値のリアルタイム表示

続いて、「ValueChanged」イベントによるタグ値のリアルタイム表示を行います。リアルタイム表示とは、タグの変化に応じた画面の動的更新の事です。

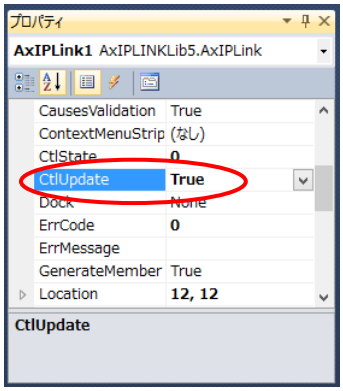
1．コントロールの貼り付け

モニタ用のテキストを以下のように配置します。



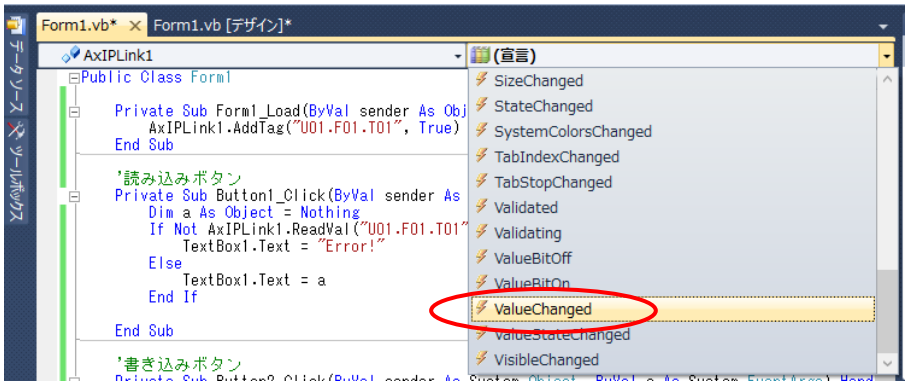
2．IPLink コントロールのプロパティ設定

CtlUpdate プロパティを True に設定してください（VB6 の場合はプロパティ名は Update になります）



3．ValueChanged イベント

IPLink コントロール  をダブルクリックし、「ValueChanged」イベントを追加してください。



4．コード記述

「ValueChanged」イベントに下記のコードを記述してください。

●VB.Net

```
Private Sub AxIPLink1_ValueChanged(ByVal sender As Object,
    ByVal e As AxIPLINKLib5._DClientIPLinkEvents_ValueChangedEvent)
        Handles AxIPLink1.ValueChanged
    Select Case e.tagPath
        Case "U01.F01.T01"
            TextBox3.Text = e.value
    End Select
End Sub
```

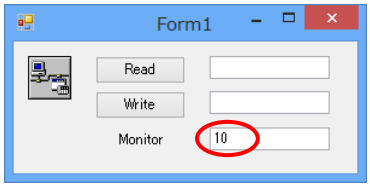
●VB6

```
Private Sub IPLink1_ValueChanged(ByVal TagPath As String, ByVal Value As Variant)
    Select Case TagPath
        Case "U01.F01.T01"
            Text3.Text = Value
    End Select
End Sub
```

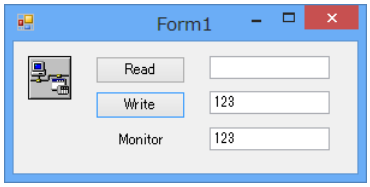
5．プログラム実行

Visual Basic の開始ボタンを押して、プログラムを開始して下さい。

フォームが表示されると、テキストコントロール（Text3）にタグの現在値が表示されます。



Text2 に任意の数値を書き込み、書き込みボタンを押してください。入力した数値がタグに書き込まれ、Text3 の表示も伴って変化することを確認して下さい。



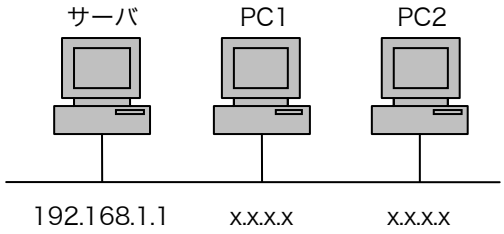
FA シリーズ側からタグの値を直接書き込むことができます。手順としては、タグ（「T01」など）を右クリックしてメニューを表示し、メニューから「タグ操作」「書き込み」を選択します。下記のダイアログが表示されるので、ダイアログから任意の値を入力します。



２－２　リモート接続

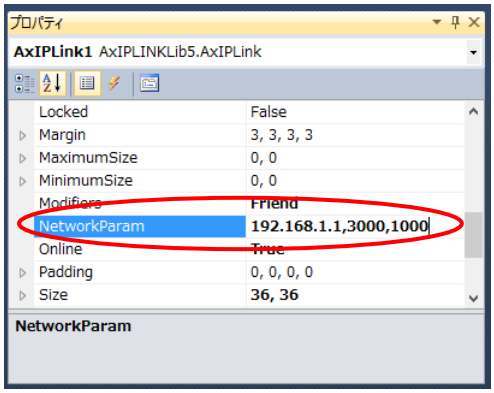
概要

IPLink サーバへの接続は、リモート PC からも簡単に行うことができます。ここでは下記構成のように、IPLink クライアントをサーバ以外の PC 上で実行させ、IPLink サーバに接続するための方法について解説します。



この例では、IPLink サーバの IP アドレスを「192.168.1.1」とします。また、クライアントは２－１で作成したアプリケーションを一箇所のみ変更して利用します。IPLink コントロールの以下のプロパティを変更して下さい。

（変更内容）
IPLink コントロールのプロパティに、IPLink サーバの IP（192.168.1.1）を設定する



NetworkParam :　「**192.168.1.1**,3000,1000」

クライアントのセットアップ

1．IPLink クライアントライブラリをクライアント PC へインストールする
それぞれのクライアント PC で、インストーラーから「IPLink クライアント」のみを選択してインストールを実行してください。

2．クライアントを実行する
各 PC 上で、クライアントアプリケーションを実行して下さい。クライアントアプリケーションは何台の PC から実行しても構いませんし、１台の PC 上で複数のクライアントアプリケーションを実行することもできます。



クライアントライブラリ（ClientIPLink5.ocx）は、以下の方法によりインストーラーを使用せずに手動でセットアップすることも可能です。

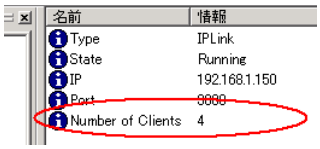
1．インストールフォルダから「ClientIPLink5.ocx」をそれぞれの PC へコピーする。
2．各クライアントのウインドウズディレクトリ内から「regsvr32.exe」を検索し、「ClientIPLink5.ocx」をドラックアンドドロップする。

但し、マイクロソフト社の共有ライブラリが不足している場合には、セットアップが正しく行えない場合があります。詳細は「インストールガイド」を参照下さい。

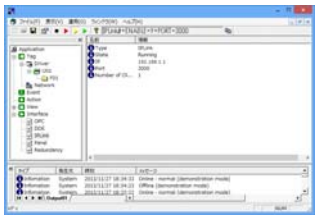
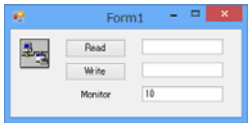
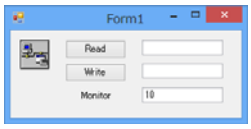
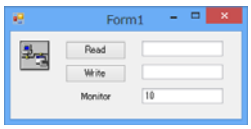
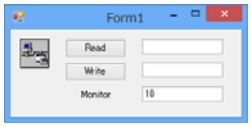
接続確認

クライアントの実行が完了したら、「Number of Clients」の値を確認してください。このカウントが、起動しているクライアントアプリケーションの数と等しければ、接続は正しく行われています。カウントが正しければ、単体での動作確認と同様の操作を行い、各クライアントからタグの読み書きが行えることを確認してください。

尚、以下の例は計４つのクライアントアプリケーションを起動しています。



「Number of Clients」が「4」になっている

PC	サーバ	クライアント	
サーバ PC			ローカル接続で１クライアント
リモート PC１	なし	 	リモート接続で２クライアント （※同じアプリケーションを２つ同時に起動している状態）
リモート PC２	なし		リモート接続で１クライアント

3章　機能解説

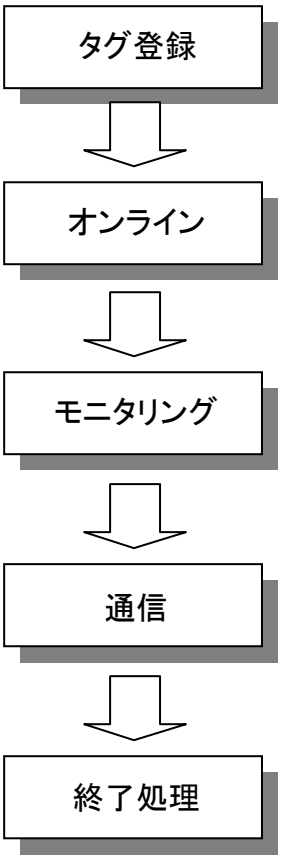
インターフェース一覧

IPLINK は、下記のインターフェースを持っています（プロパティ、メソッド、イベント)。各インターフェースの詳細については「5章　リファレンス」を参照して下さい。

メソッド・プロパティ	
■接続処理	
NetworkParam	IPLink サーバとの接続パラメータ。
Online	IPLink サーバと接続/切断を行う。
Update	タグの自動更新を開始/終了する。
■タグ登録・解除	
AddTag	タグを登録する。
AddTags	複数のタグを一括で登録する。
ActiveTag	タグのアクティブ状態を変更する。
ActiveTags	複数のタグのアクティブ状態を一括で変更する。
DelTag	タグの登録を解除する。
DelTags	複数のタグの登録を一括で解除する。
■通信	
ReadVal	タグの値を読み込む。
ReadVals	複数のタグの値を一括で読み込む。
WriteVal	タグに値を書き込む。
WriteVals	複数のタグに値を一括で書き込む。
■その他	
AccessPath	タグの共通パス名。
State	IPLink サーバとの接続状態（0-NG 1-OK）
ErrMessage	最後に発生したエラーのメッセージ内容。
イベント	
ValueStateChanged	値及び値品質が変化したときに発生する。
ValueChanged	値が変化したときに発生する。
ValueBitOn	ビットが On したときに発生する。
ValueBitOff	ビットが Off したときに発生する。
StateChanged	IPLink サーバの生存状態が変化したときに発生する。

一般的なプログラムの処理フロー

一般的なプログラミングによる処理の流れは以下のとおりです。



AddTag

通信を行うタグの登録を宣言します。
タグの登録はオンライン後でも可能です。

NetworkParam
Online = True

オンラインが True の間、サーバとの通信が行えます。

Update = True
ValueChanged

Update プロパティを True にすると、自動更新を開始します。
タグ値が変化した時に ValueChanged イベントが発生します。
値変化による動的な処理の実装はこのイベントを利用します。

ReadVal
WriteVal

メソッドを利用したタグの読み書きを行います。

Update =false
Online = false

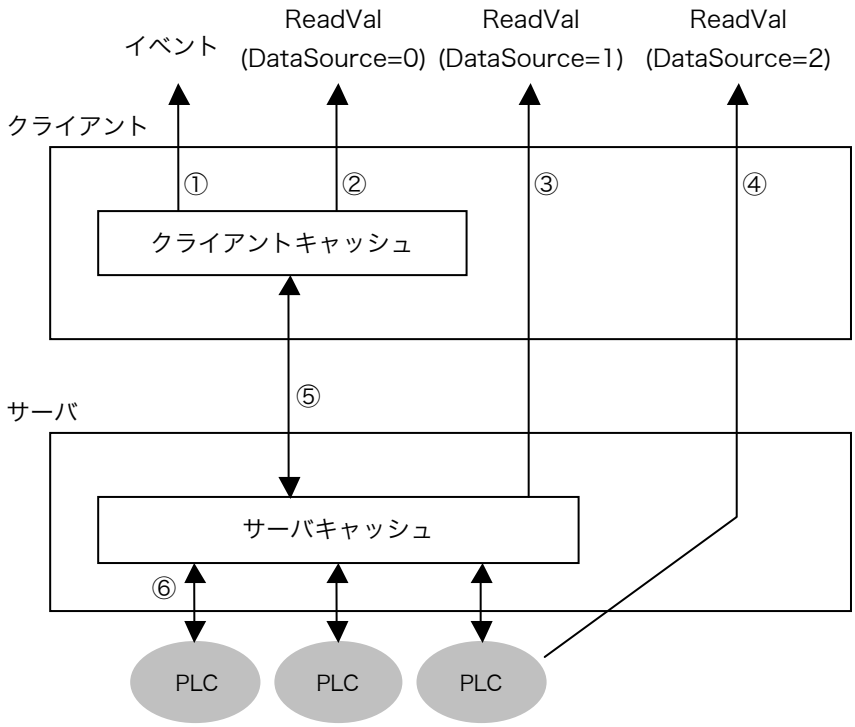
通信の終了処理はフォームを閉じると自動的に行われます。
従って、通常は終了処理を意識する必要はありません。

クライアントキャッシュとサーバキャッシュ

データの読み込みには、下記の４つの方法があります。

① イベントを利用・・・・・・・・・・・・・・・・・・・・・・ ValueChanged など
② クライアントキャッシュを利用・・・・・・・・ ReadVal(DataSource=0)
③ サーバキャッシュを利用・・・・・・・・・・・・・ ReadVal(DataSource=1)
④ 直接通信・・・・・・・・・・・・・・・・・・・・・・・・・・・ ReadVal(DataSource=2)

クライアント（ClientIPLink5.ocx）及びサーバにはそれぞれ独立したキャッシュが存在します。サーバキャッシュの更新⑥は AddTag によりアクティブとして登録されたタグのみ行われます。アクティブなタグは③ReadVal(DataSource=1)での読み込みが可能です。アクティブでないタグはサーバキャッシュが更新されないため、④ReadVal(DataSource=2)での読み込みのみ有効です。クライアントキャッシュの更新⑤は Update プロパティが True の時、実行されます。更新中は、値が変化すると各イベント①が発生し、非同期な値の取得ができます。また、②ReadVal(DataSource=0)により、クライアントキャッシュから値を直接読み込むことができます。
サーバキャッシュの更新⑥の周期はサーバ側のタグ設定のフォルダプロパティによって行い、クライアントキャッシュの更新⑤の周期は NetworkParam で指定します。



下表は、各読み込み方法の条件と特徴です。

通信方法	特徴	アクティブ	Update
イベント ・モニタリング時、推奨	イベントでの値の取得は、モニタリングに適しています。クライアントキャッシュを自動更新するための通信は、Visual Basic とは異なるタスクで実行されるため、Visual Basic の動作を妨げません。但し、常にサーバから定周期で値の取得が行われるため、タグの点数が多くなるとクライアント、サーバ間の通信負荷が高くなる可能性があります。そのような場合は更新周期を遅くする方向で調整を行って下さい。	True (必須)	True (必須)
クライアントキャッシュ値 ReadVal(DataSource=0) ・通常時、推奨	クライアントキャッシュからの値取得は、ReadVal の処理速度が最も速く処理される点が特徴です。Visual Basic 上で迅速な ReadVal の反応が求められるような場合には、この方法による取得をお奨めします。	True (必須)	True (必須)
サーバキャッシュ値 ReadVal(DataSource=1) ・イベントなし時、推奨	サーバキャッシュからの値取得は、クライアント・サーバ間の自動更新が不要であるため、クライアント・サーバ間のネットワーク負荷を下げる効果があります。但し、値変化によるイベントは発生しません。	True (必須)	False (推奨)
デバイス値 ReadVal(DataSource=2) ・制御、バッチ処理時推奨	この方法は、最も処理時間がかかる値の取得方法です。サーバ対して強制的に PLC との通信を要求するため、他の通信に影響を与える可能性があります。 但し、要求の都度、実際に PLC と通信が行われた結果が返ってくるため、最も新しいデバイス値を取得することができます。また、タグ数が大量かつ通信頻度が低いような場合には、サーバ・PLC 間の通信負荷を下げる効果を得ることができます。従って、この方法は PLC との同期が必要な制御の場合や、バッチ的な通信に最適です。	False (推奨)	False (推奨)

これら４つの通信方法をシステム要件にあわせ選択し、時には組み合わせることにより、通信パフォーマンスを高めることができます。反対に、組み合わせ次第ではパフォーマンスの低下をまねくことがある点に注意してください。例えば、タグをアクティブで登録し、Update を True、ReadVal(DataSource=2)で通信を行うといった組み合わせは最もパフォーマンスが低下する組み合わせです。



これらの通信方法を特に意識したくない場合には、以下の設定の組み合わせを推奨します。

●すべてのタグをアクティブにする。

●Update を True にする。

●ReadVal(DataSource=0) 及び ValueChanged イベントで値を取得する。

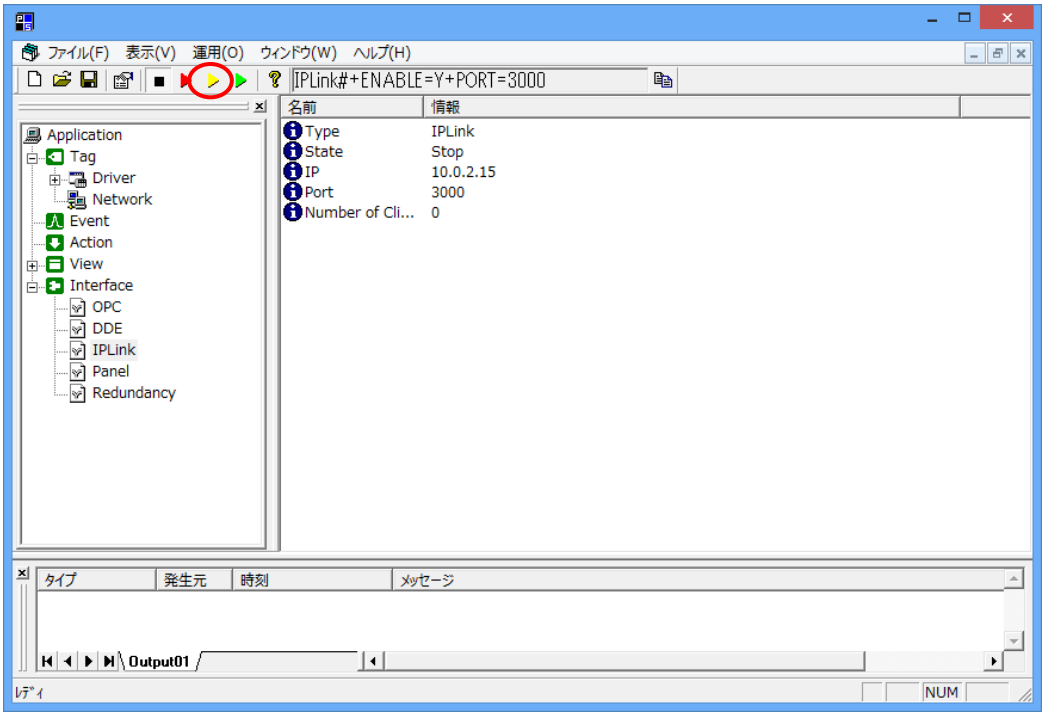
4章　Visual Basic による IPLink 接続の実践

概要

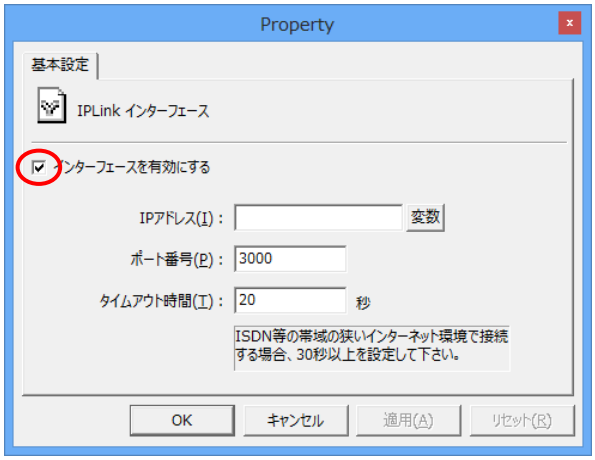
この章では、Visual Basic による IPLink 接続アプリケーションのサンプルを紹介します。
サンプルの動作は以下のように行ってください。ここでは VB.Net2010 の Sample01 を例に動作を行います。

サンプルは以下に格納されています。
¥sample¥common¥interface_iplink¥VS2010¥sample01

1. Server アプリケーションを起動し、設定ファイル(vs2010_sample01.txt)を読み込みオンラインにする。

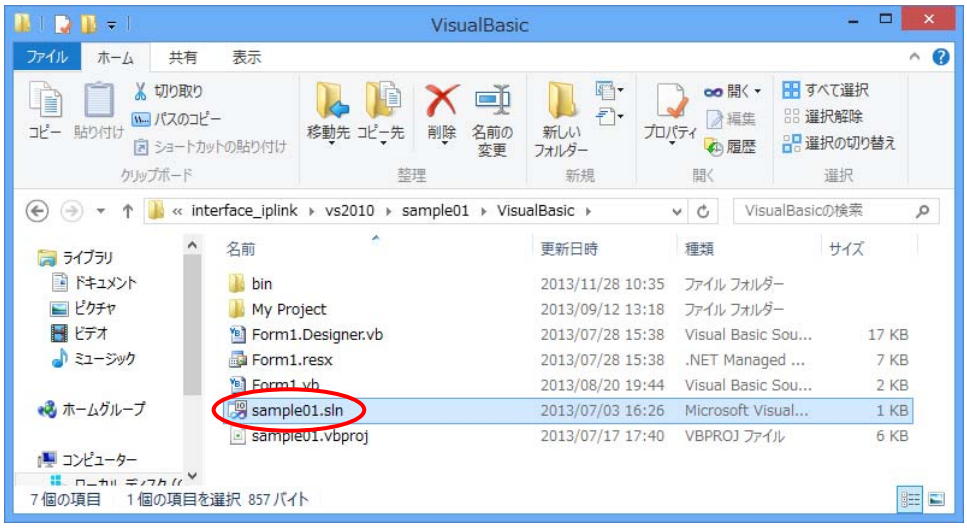


尚、IPLink インターフェースが有効になっている事を確認してください。

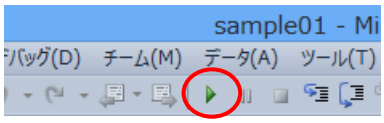


2. Visual Studio2010 を起動する。

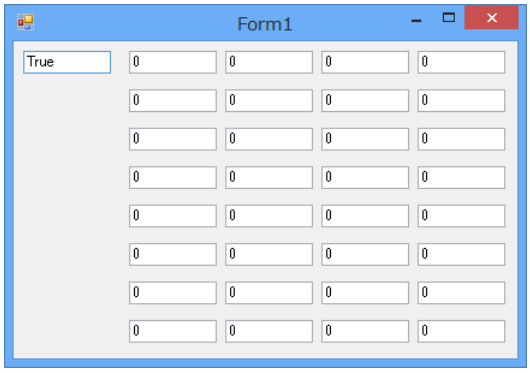
¥¥sample¥common¥interface_iplink¥VS2010¥sample01¥VisualBasic¥sample01.sln を実行します。



3. Visual Studio2010 が起動したら、デバッグ開始ボタンを押下し、アプリケーションを実行します。



4. アプリケーションが正常に動作することを確認



注意

開発やプログラムコードを参照する場合は、サンプルのバージョン(VB6.0～VS2010)に該当する Visual Studio を事前にインストールしておく必要があります。

ヒント

本項で紹介する各種サンプルのプログラムコードは、Visual Studio を起動していただき、ご確認ください。

ヒント

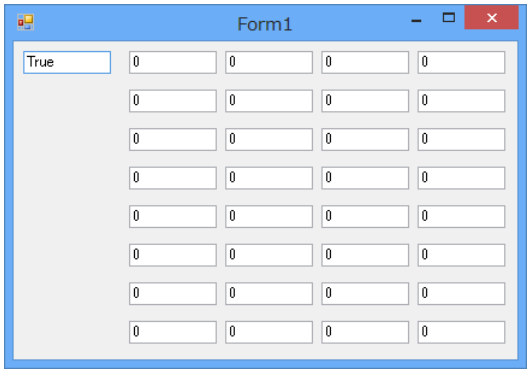
Visual Studio をインストールしていない場合でも動作を確認することが出来ます。その場合は、¥bin フォルダ内の sampleXX.exe を実行してください。

４－１　モニタリング（Sample01）

概要

このサンプルは、３２点のデータをリアルタイムにモニタリングするサンプルです。タグをアクティブとして登録し、Update プロパティを True とすることにより、値変化時にイベントが発生します。

各項目の表示については、StateChanged イベントを使用し、IPLink サーバとの接続状況の表示を行っています。また、ValueStateChanged イベントを使用し、タグの値及び品質が変化した場合、各タグ値の表示を自動的に更新します。



このサンプルは、以下に格納されています。
¥¥sample¥common¥interface_iplink¥vb6～vs2010¥sample01

プログラムの要点

・フォームロード処理

フォームロード時に、IPLink サーバと接続を行うための NetworkParam プロパティの設定を行います。合わせて通信を行うタグを AddTag で登録を行うなどといった初期処理を行います。

●VB.Net

```
AxIPLink1.AddTags(32, TagPaths, Actives, Results)
AxIPLink1.NetworkParam = ",3000,1000"
AxIPLink1.Online = True
AxIPLink1.CtlUpdate = True
```

●VB6.0

```
IPLink1.AddTags 32, TagPaths, Actives, Results
IPLink1.NetworkParam = ",3000,1000"
IPLink1.Online = True
IPLink1.Update = True
```



クライアントとサーバ間の値の更新周期は NetworkParam で設定します。サーバと PLC 間の値の更新周期は各 PLC のフォルダプロパティで設定します。

・StateChanged イベント

StateChanged は IPLink サーバの接続状態が変化した時に発生します。接続中であれば「True」、未接続であれば「False」が取得できます。ここでは、取得した接続状態を、エディットボックスへ表示しています。

●VB.Net

```
Private Sub AxIPLink1_StateChanged(ByVal sender As Object,
    ByVal e As AxIPLINKLib5._DClientIPLinkEvents_StateChangedEvent)
        Handles AxIPLink.StateChanged
    txtState.Text = CStr(e.state)
End Sub
```

●VB6.0

```
Private Sub IPLink1_StateChanged(ByVal State As Boolean)
    txtState.Text = State
End Sub
```

・ValueStateChanged イベント

ValueStateChanged は IPLink サーバのタグの値及び品質が変化したときに発生します。ここでは、タグの品質が正常（True）であった場合、値のエディットボックスへ表示します。また、品質が異常になった場合、「Error!」と表示させています。

●VB.Net

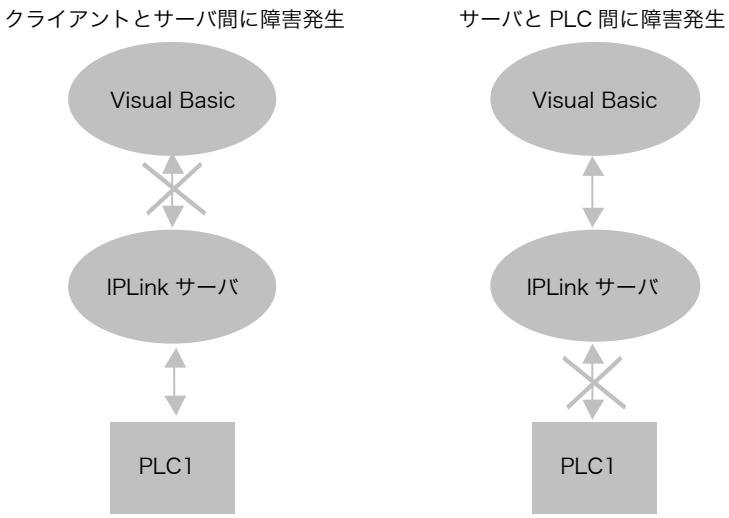
```
Private Sub AxIPLink1_ValueStateChanged(ByVal sender As Object,
    ByVal e As AxIPLINKLib5._DClientIPLinkEvents_ValueStateChangedEvent)
        Handles AxIPLink.ValueStateChanged
     '正常時
    If e.state Then
        txtVal(Integer.Parse(e.tagPath.Substring(e.tagPath.Length - 2))).Text = CStr(e.value)
     '異常時
    Else
        txtVal(Integer.Parse(e.tagPath.Substring(e.tagPath.Length - 2))).Text = "Error!"
    End If
End Sub
```

●VB6.0

```
Private Sub IPLink1_ValueStateChanged(ByVal TagPath As String,
ByVal Value As Variant, ByVal State As Boolean)
     '正常時
    If State Then
        txtValue(CLng(Right(TagPath, 2))).Text = Value
     '異常時
    Else
        txtValue(CLng(Right(TagPath, 2))).Text = "Error!"
    End If
End Sub
```

通信切断の検証

通信切断の原因としては、大きく以下の２つに分けられます。



通信障害の発生時、このサンプルで使用している２つのイベントは下記のように動作します。

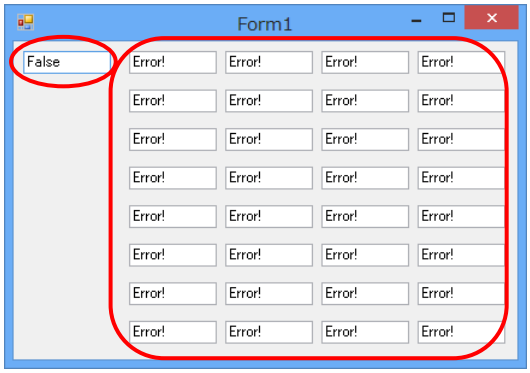
■ValueStateChanged(ByVal TagPath As String, ByVal Value As Variant, ByVal State As Boolean)
「ValueStateChanged」イベントは３つの引数があり、第３引数である「State」は値の品質を表します。何らかの理由で値が取得できなかったとき、State 引数には False が設定されます。これは、クライアントとサーバ間の障害、サーバと PLC 間の障害、いずれの障害においても同様に発生します。

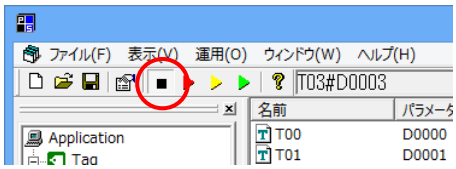
■StateChanged(ByVal State As Boolean)
「StateChanged」イベントは、クライアントとサーバ間の通信状態が変化した時に発生するイベントです。サーバと PLC 間で通信ができなくなった時は発生しません。障害時には「State」引数に False が設定されます。

このサンプルを使用して、通信障害時の動作検証を行います。

1．クライアントとサーバ間の通信障害

クライアントとサーバ間の通信障害を模擬するため、正常通信中にアプリケーションをオフラインにして下さい。すると、StateChanged 及び ValueStateChanged が発生します。フォーム上では値が「Error!」になり、状態が「False」と表示されることを確認してください。

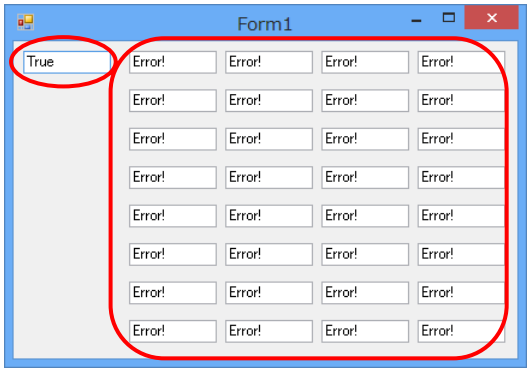


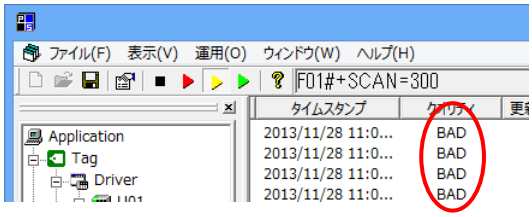


2．サーバと PLC 間の通信障害

サーバと PLC 間の通信障害を模擬するため、正常通信中に U01 ユニットを右クリックし、「ユニット操作」「切断」を選択して下さい。切断操作により、PLC との通信不能状態をシミュレートすることができます。

この場合、ValueStateChanged のみ発生します。フォーム上では値が「Error!」になりますが、状態は「True」のまま表示されています。





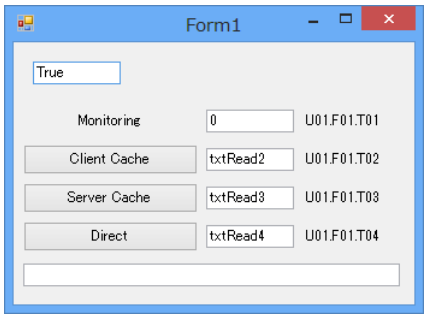
４－２　データの読み込み（Sample02）

概要

このサンプルは、３章の「クライアントキャッシュとサーバキャッシュ」で解説した４通りの方法によってタグの値を読み込むサンプルアプリケーションです。

① イベントを利用・・・・・・・・・・・・・・・・・・・・・・ValueChanged
② クライアントキャッシュを利用・・・・・・・・ReadVal(DataSource=0)
③ サーバキャッシュを利用・・・・・・・・・・・・・ReadVal(DataSource=1)
④ 直接通信・・・・・・・・・・・・・・・・・・・・・・・・・・ReadVal(DataSource=2)

の４種類のデータ取得方法となります。
イベントは ValueChanged イベントを使用し、他処理については ReadVal メソッドを使用します。



このサンプルは、以下に格納されています。
¥¥sample¥common¥interface_iplink¥vb6～vs2010¥sample02



クライアントキャッシュとサーバキャッシュの違いは、章の「クライアントキャッシュとサーバキャッシュ」を参照してください。

プログラムの要点

・ValueChanged イベント

ValueChanged イベントはタグの値が変化すると発生するイベントです。戻り値「Value」には通信が成功した場合の値が格納されているため、信頼できる値となります。

●VB.Net

```
Private Sub AxIPLink1_ValueChanged(ByVal sender As Object,
    ByVal e As AxIPLINKLib5._DClientIPLinkEvents_ValueChangedEvent) Handles IPLink1.ValueChanged
    Select Case e.tagPath
        Case "U01.F01.T01"
            txtRead1.Text = e.value
    End Select
End Sub
```

●VB6.0

```
Private Sub IPLink1_ValueChanged(ByVal TagPath As String, ByVal Value As Variant)
    Select Case TagPath
        Case "U01.F01.T01"
            txtRead1.Text = Value
    End Select
End Sub
```

・ReadVal(DataSource=0)

ReadVal メソッドの第三引数 DataSource を 0 とすると、クライアントキャッシュの値を取得します。この方法は最も高速なデータ取得方法となるため、迅速な ReadVal の反応が求められる場合は、この方法を使用します。

●VB.Net

```
If Not AxIPLink1.ReadVal("U01.F01.T02", a, 0) Then
    txtError.Text = IPLink1.ErrMessage
    Exit Sub
End If
```

●VB6.0

```
If Not IPLink1.ReadVal("U01.F01.T02", a, 0) Then
    txtError.Text = IPLink1.ErrMessage
    Exit Sub
End If
```

・ReadVal(DataSource=1)

ReadVal メソッドの第三引数 DataSource を 1 とすると、サーバキャッシュの値を取得します。サーバ側で更新されている値を取得するため、クライアント側とサーバ間で定期的な自動更新が不要になるため、その間の負荷を下げる効果があります。そのため、ネットワークの帯域などが狭い場合などに使用する方法になります。

●VB.Net

```
If Not AxIPLink1.ReadVal("U01.F01.T03", a, 1) Then
    txtError.Text = IPLink1.ErrMessage
    Exit Sub
End If
```

●VB6.0

```
If Not IPLink1.ReadVal("U01.F01.T03", a, 1) Then
    txtError.Text = IPLink1.ErrMessage
    Exit Sub
End If
```

・ReadVal(DataSource=3)

ReadVal メソッドの第三引数 DataSource を 3 とすると、デバイス値を取得します。この方法は、強制的に PLC との通信要求を行うため、もっとも処理時間と負荷が高い方法になります。
ただし、要求の都度、実際に PLC と通信が行われた結果が帰ってくるため、最も新しいデバイス値を取得することができます。PLC との同期が必要な場合や低周期的なデータ取得が必要なく、バッチ処理のような一定の期間（間隔）ごとに処理をおこなう場合はこちらの方法が最適です。

●VB.Net

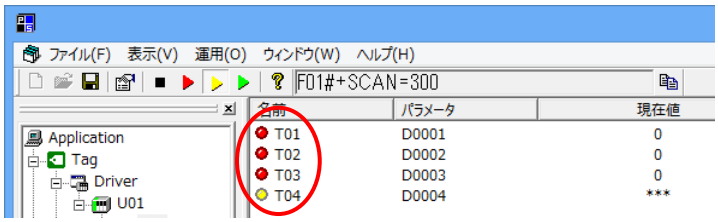
```
If Not AxIPLink1.ReadVal("U01.F01.T04", a, 2) Then
    txtError.Text = IPLink1.ErrMessage
    Exit Sub
End If
```

●VB6.0

```
If Not IPLink1.ReadVal("U01.F01.T04", a, 2) Then
    txtError.Text = IPLink1.ErrMessage
    Exit Sub
End If
```

動作確認

オンライン後、Visual Basic アプリケーションを実行して下さい。
タグのランプが「T01」～「T03」まで赤色、「T04」が黄色になったのを確認してください。



通信接続が正しく行われていることを確認したら、各読み込みボタンを押してタグの値が正しく読み込まれることを確認してください。

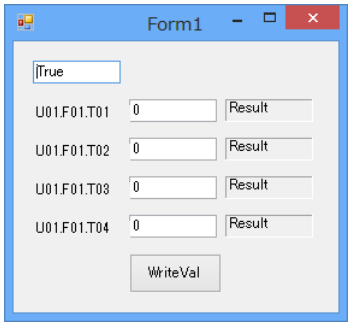
「T04」は「AddTag」でアクティブでない状態で登録したため、サーバと PLC 間で自動更新が行われません。アクティブでないタグはランプが黄色になります。

IPLink1.AddTag "U01.F01.T04", **False**

４－３　データの書き込み（Sample03）

概要

このサンプルは、値の書き込みを行うサンプルプログラムです。複数のタグに対して、値を書き込む場合 WriteVals メソッドが最適です。このサンプルは、WriteVals を使用し、４つのタグに対して同時に書き込み処理を行います。



このサンプルは、以下に格納されています。
¥¥sample¥common¥interface_iplink¥vb6～vs2010¥sample03

プログラムの要点

・WriteVals メソッド

WriteVals では、複数のタグをまとめて処理することができます。複数の連続したアドレス（D0000 から D01000 など）のタグの書き込みがある場合、WriteVal を複数回呼び出すよりも、WriteVals によって一括して書き込みを行った方が、通信パケットがまとめて処理されるため、高速な通信が行われます。また、大量な点数を扱う場合には、配列タグを用いる事によって高速に書き込みを行う事ができます。

サンプルの「btnWrite_Click()」では、WriteVals 返値で成功したかどうかの判別を行っています。また、Results 引数にはタグごとに成否のブール値が配列で格納されますが、これは、WriteVals 返値が TRUE の時のみ使用することができます。



WriteVals での戻り値 Values は必ずオブジェクト型（VB6.0 はバリアント型）配列で宣言して下さい。これは、書き込む値に複数の異なる型を混在させて処理させるようにする為です（ブール型、数値型、文字列型、ブール配列型、数値配列型、文字列配列型など）。

●VB.Net

```
If IPLink1.WriteVals(4, TagPaths, TxtValues, Results) Then
    For i = 0 To 3
        If Results(i) Then
            Controls("lblResult" & i).Text = "Success!"
        Else
            Controls("lblResult" & i).Text = "Error!"
        End If
    Next i
Else
    For i = 0 To 3
        Controls("lblResult" & i).Text = "Error!"
    Next i
End If
```

●VB.6.0

```
If IPLink1.WriteVals(4, TagPaths, Values, Results) Then
    For i = 0 To 3
        If Results(i) Then
            lblResult(i).Caption = "Success!"
        Else
            lblResult(i).Caption = "Error!"
        End If
    Next i
Else
    For i = 0 To 3
        lblResult(i).Caption = "Error!"
    Next i
End If
```

動作確認

オンライン後、Visual Basic アプリケーションを実行して下さい。

「T01」～「T04」のタグのランプが黄色になる事を確認してください（この例は書き込みのみが目的であるため（値を自動更新する必要がないため、アクティブでない状態で登録しています)。現在の値を確認したい場合には、タグを右クリックしてメニューを表示し、メニューから「タグ操作」「全タグモニタリング開始」を選択する事で、現在値を確認できるようになります（ランプが赤色になる)。



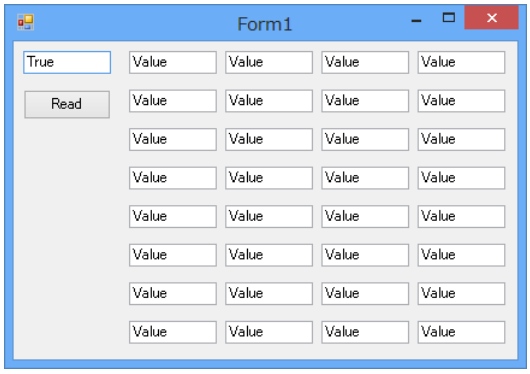
通信接続を確認後、Visual Basic 画面から任意の値を入力し、書き込みボタンによってタグに値が書き込まれることを確認してください。

４－４　大量なデータの扱い方（Sample04）

概要

配列タグを利用したサンプルです。配列タグを用いると、大量のデータを１つのタグにまとめて通信することができます。

タグを配列タグとして設定を行うと、連続したエリアをまとめて扱うことができます。
例えば、PLC のアドレス「D00001」の数値タグで、配列数を 1000 と指定した場合、「D00001」から「D01000」までの 1000 個の配列として扱えます



このサンプルは、以下に格納されています。
¥¥sample¥common¥interface_iplink¥vb6～vs2010¥sample04



配列タグの詳細については「タグマニュアル」を参照して下さい



IPLink クライアントでは ReadVal/ WriteVal などで配列を扱うこともできます。

プログラムの要点

・タグの登録

AddTag メソッドでタグの登録を行います。このサンプルではタグを配列化しており、1 つのタグとして扱うことができるため、AddTag を行う対象も 1 つのタグのみになり、複数タグに対しての AddTag を行う必要はありません。

●VB.Net

```
AxIPLink1.AddTag("U01.F01.T01", False)
```

●VB6.0

```
IPLink1.AddTag("U01.F01.T01", False)
```

・データ取得

このサンプルではデータ取得を ReadVal メソッドで行っています。配列タグを指定して ReadVal を行った場合、戻り値 Value は配列のデータとして帰ってきます。

●VB.Net

```
If AxIPLink1.ReadVal("U01.F01.T01", Values, 2) Then
    For i = 0 To 31
        Controls("txtValue" & i).Text = Values(i)
    Next i
Else
    For i = 0 To 31
        Controls("txtValue" & i).Text = "Error!"
    Next i
End If
```

●VB.Net

```
If IPLink1.ReadVal("U01.F01.T01", Values, 2) Then
    For i = 0 To 31
        txtValue(i).Text = Values(i)
    Next i
Else
    For i = 0 To 31
        txtValue(i).Text = "Error!"
    Next i
End If
```



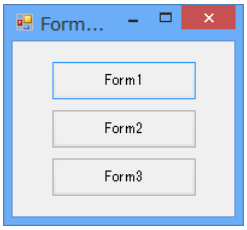
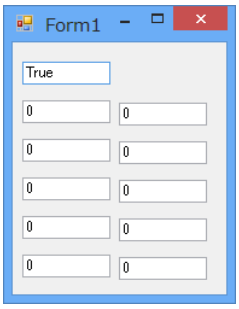
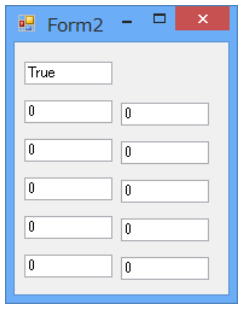
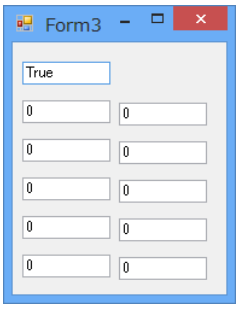
ReadVal での戻り値 Value は必ずオブジェクト型（VB6.0 はバリアント型）配列で宣言して下さい。これは、読み込んだ値に複数の異なる型を混在させて処理させるようにする為です（ブール型、数値型、文字列型、ブール配列型、数値配列型、文字列配列型など）。

４－５　複数フォームでの扱い方（Sample05）

概要

IPLink サーバは、複数の IPLink クライアントから同時に接続することが可能です。このサンプルでは、同じプロジェクト内の複数のフォームから同時に接続を行います。

複数のフォームから通信を行う場合には、各フォーム上に「IPLink」コントロールを貼り付けます。フォームを閉じると通信切断処理が自動的に行われ、サーバへ登録したタグも解除されます。尚、１プロジェクト内に複数のフォームを作成するのと同様に、複数のプロジェクトから同時に通信を行うことも可能です。

FormMain	Form1	Form2	Form3
			

このサンプルは、以下に格納されています。
¥¥sample¥common¥interface_iplink¥vb6～vs2010¥sample05

プログラムの要点

接続状態は StateChanged イベントで取得を行い、値の変化は ValueChanged イベントで取得を行います。

●VB.Net

```
Private Sub AxIPLink1_StateChanged(ByVal sender As Object,
    ByVal e As AxIPLINKLib5._DClientIPLinkEvents_StateChangedEvent) Handles IPLink1.StateChanged
    txtState.Text = CStr(e.state)
End Sub

Private Sub AxIPLink1_ValueChanged(ByVal sender As Object,
    ByVal e As AxIPLINKLib5._DClientIPLinkEvents_ValueChangedEvent) Handles IPLink1.ValueChanged
    txtVal(Integer.Parse(e.tagPath.Substring(e.tagPath.Length - 2))).Text = e.value
End Sub
```

●VB6.0

```
Private Sub IPLink1_StateChanged(ByVal State As Boolean)
    txtState.Text = State
End Sub

Private Sub IPLink1_ValueChanged(ByVal TagPath As String, ByVal Value As Variant)
    txtValue(CLng(Right(TagPath, 2))).Text = Value
End Sub
```

動作確認

1．アプリケーションの実行
オンライン後、Visual Basic アプリケーションを実行して下さい。ここで、全てのランプが黒色であることを確認して下さい。

2．「Form1」の表示
「Form1」ボタン押して画面を表示して下さい。Form1 が表示され、画面上に値が表示されることを確認して下さい。この時、「T00」～「T09」までのランプは赤になり、「T10」～「T29」までのランプは黒のままであることを確認して下さい。

3．「Form1」を閉じる
「Form1」を閉じて、全てのランプが黒色になることを確認して下さい。これは、「Form1」が閉じられた事によって接続が切れ、タグの登録が解除された事を意味します（サーバと PLC 間の通信も停止します）。

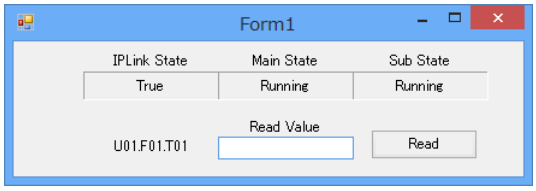
4．全てのフォームを表示する
「Form1」「Form2」「Form3」ボタン押して、全ての画面を表示して下さい。画面が表示されると、全のランプが赤色になることを確認して下さい。

４－６　二重化運用サーバへの接続（Sample06）

概要

Server アプリケーションを二重化運用している時に、ランニングしている Server アプリケーションへ接続を行うサンプルです。

このサンプルでは、IPLink サーバとの接続が切れた場合、他の IPLink サーバへ接続を自動的に行う動作をします。この動作により、Server アプリケーションの二重化運用に対応します。



このサンプルは、以下に格納されています。
¥¥sample¥common¥interface_iplink¥vb6～vs2010¥sample06



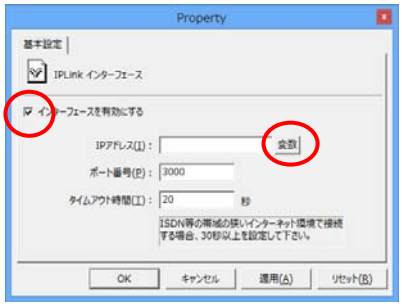
二重化運用の詳細な内容や設定方法については、二重化マニュアルを参照してください。



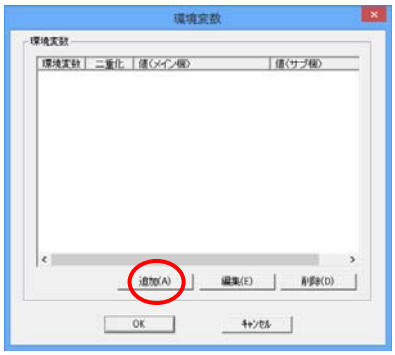
本サプルではメイン機側 IP アドレス：192.168.1.1、サブ機側 IP アドレス：192.168.1.2 としています。
実際に動作などを行う際には、IP アドレスを環境に合わせて変更してください。

サーバ設定

IPLink インターフェースが有効になっている事を確認してください。また、メインとサブの IP アドレスを設定します。本サンプルでは、環境変数を使用して、IP アドレの設定を行います。変数ボタンを押して、設定画面を表示します。

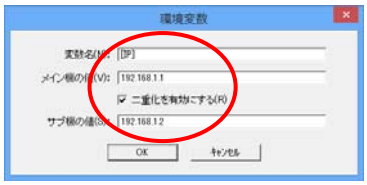


環境変数設定画面が表示されたら、追加ボタンを押します。



以下のように設定を行います。

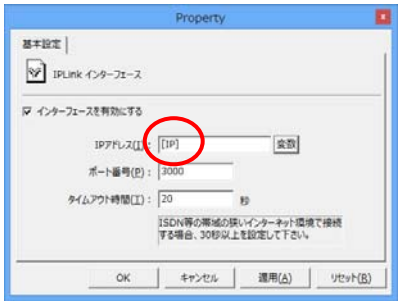
変数名：[IP]
メイン機の値：192.168.1.1
二重化を有効にするにチェック
メイン機の値：192.168.1.2



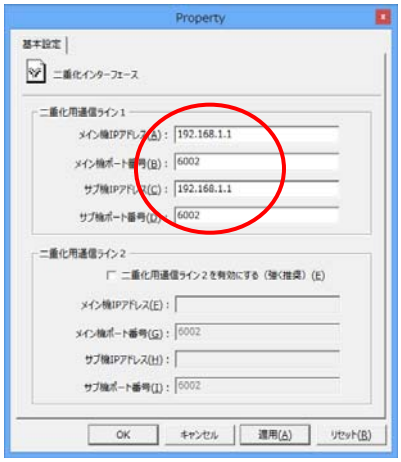
OK ボタンを押して、ダイアログを閉じ、追加した環境変数を選択し、OK ボタンを押します。



IP アドレス項目に環境変数が設定されていることを確認し、OK ボタンでダイアログを閉じます。



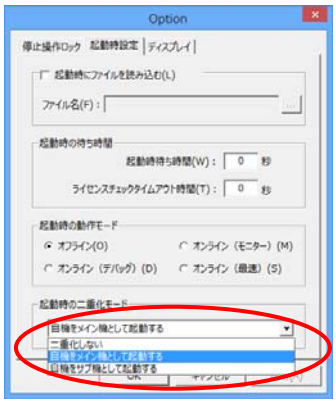
次に Server アプリケーションの二重化運用設定を行います。
Redundancy インターフェースのプロパティを開き、メインとサブそれぞれの IP アドレスとポート番号の設定を行います。設定完了後、OK ボタンでダイアログを閉じます。



Server アプリケーションをメイン、サブどちらで動作させるかを設定します。本設定はメイン機及びサブ機の両方で行う必要があります。

「運用」-「オプション」-「起動時設定」を開き、設定を行います。
メイン機の場合は「自機をメイン機として起動する」
サブ機の場合は「自機をサブ機として起動する」
を選択します。

この設定は、Server アプリケーション再起動後から有効になるため、設定完了後、Server アプリケーションを再起動します。





通常はメイン機側/サブ機側で IP アドレスが異なるため、それぞれの環境に合わせた IP を、インターフェースに設定する必要があります。そうすると、サーバアプリケーションの設定ファイルが複数必要になります。その場合、複数の設定ファイルの管理を行う必要性が出てきます。
これを避けるため、環境変数機能を使用します。この機能を使用することで、メインン機側/サブ機側どちらの環境でも 1 つの設定ファイルで動作できるようになります。

プログラム

・定数の設定

このサンプルでは定数として、メイン機側とサブ機側へ接続する際の、それぞれの NetworkParam を定数として定義してあります。この値を変更することで、接続先を任意のものに変更することが可能です。

```
'メイン機ネットワークパラメータ（IP,ポート番号,更新周期,通信タイムアウト,接続タイムアウト）
Private Const fcNetworkParam1 As String = "192.168.1.1, 3000, 1000, 3, 5"
'サブ機ネットワークパラメータ（IP,ポート番号,更新周期,通信タイムアウト,接続タイムアウト）
Private Const fcNetworkParam2 As String = "192.168.1.2, 3000, 1000, 3, 5"
```

・フォームロード処理

フォームロード時に NetworkParam プロパティの設定や、AddTag などの初期設定を行います。
AddTag には通常のタグの他に、「//Redundancy/StateMain」と「"//Redundancy/StateSub"」を行います。
この 2 つはシステムプロパティと呼ばれるもので、Server アプリケーションの二重化運用時のステータスを取得できるプロパティです。

システムプロパティの値を取得する場合は、一般的なタグと同様に AddTag する必要が有るため、ここの処理で AddTag を行います。

●VB.Net

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    Dim i As Integer
    Dim vResults As Object = Nothing
    Timer1.Enabled = False
    fvSystemTags(0) = "//Redundancy/StateMain"
    fvSystemTags(1) = "//Redundancy/StateSub"

    For i = 0 To 2
        If (i < 1) Then
            fvTagName(i) = "U01.F01.T01"
        Else
            fvTagName(i) = fvSystemTags(i - 1)
        End If
        fvActives(i) = True
    Next i

    AxIPLink1.AddTags(3, fvTagName, fvActives, vResults)
    AxIPLink1.NetworkParam = fcNetworkParam1
    AxIPLink1.Online = True
    AxIPLink1.CtlUpdate = True
End Sub
```

●VB6.0

```
Private Sub Form_Load()
    Dim i As Integer
    Dim vResults As Variant
    Timer1.Enabled = False
    fvSystemTags(0) = "//Redundancy/StateMain"
    fvSystemTags(1) = "//Redundancy/StateSub"

    For i = 0 To 2
        If (i < 1) Then
            fvTagName(i) = "U01.F01.T01"
        Else
            fvTagName(i) = fvSystemTags(i - 1)
        End If
        fvActives(i) = True
    Next i

    IPLink1.AddTags 3, fvTagName, fvActives, vResults
    IPLink1.NetworkParam = fcNetworkParam1
    IPLink1.Online = True
    IPLink1.Update = True
End Sub
```



システムプロパティで取得できる内容は以下のとおりです。

「//Redundancy/StateMain」：メイン機の状態を表します（1：異常/2：待機中/3：運転中）
「//Redundancy/StateSub」：サブ機の状態を表します（1：異常/2：待機中/3：運転中）

詳細は二重化マニュアルの「二重化ステータス」項目を参照してください。

・Server アプリケーションの二重化状態を取得

二重化システムプロパティの値を ValueStateChanged イベントで取得して、テキストボックスに表示を行います。
メイン機側とサブ機側両方の状態を画面上で確認できるようにします。

●VB.Net

```
Private Sub IPLink1_ValueStateChanged(ByVal sender As Object,
    ByVal e As AxIPLINKLib5._DClientIPLinkEvents_ValueStateChangedEvent)
        Handles IPLink1.ValueStateChanged

    '/// Redundancy Status ////////////////////////////////////
    If (fvSystemTags(0) = e.tagPath Or fvSystemTags(1) = e.tagPath) Then
        '/// Redundancy Status(Main Machine Status) ////////////////////////////////////
        If (fvSystemTags(0) = e.tagPath) Then
            If (e.value = 1) Then
                lblMainState.Text = "Abnormal"
            ElseIf (e.value = 2) Then
                lblMainState.Text = "Waiting"
            ElseIf (e.value = 3) Then
                lblMainState.Text = "Running"
            Else
                lblMainState.Text = "Unidentified"
            End If
        End If
        '/// Redundancy Status(Sub Machine Status) ////////////////////////////////////
        If (fvSystemTags(1) = e.tagPath) Then
            If (e.value = 1) Then
                lblSubState.Text = "Abnormal"
            ElseIf (e.value = 2) Then
                lblSubState.Text = "Waiting"
            ElseIf (e.value = 3) Then
                lblSubState.Text = "Running"
            Else
                lblSubState.Text = "Unidentified"
            End If
        End If
    End If
End Sub
```

●VB6.0

```
Private Sub IPLink1_ValueStateChanged(ByVal TagPath As String,
    ByVal Value As Variant, ByVal State As Boolean)

    '/// Redundancy Status ////////////////////////////////////
    If (fvSystemTags(0) = TagPath Or _
        fvSystemTags(1) = TagPath) Then
        '/// Redundancy Status(Main Machine Status) ////////////////////////////////////
        If (fvSystemTags(0) = TagPath) Then
            If (Value = 1) Then
                lblMainState.Caption = "Abnormal"
            ElseIf (Value = 2) Then
                lblMainState.Caption = "Waiting"
            ElseIf (Value = 3) Then
                lblMainState.Caption = "Running"
            Else
                lblMainState.Caption = "Unidentified"
            End If
        End If
        '/// Redundancy Status(Sub Machine Status) ////////////////////////////////////
        If (fvSystemTags(1) = TagPath) Then
            If (Value = 1) Then
                lblSubState.Caption = "Abnormal"
            ElseIf (Value = 2) Then
                lblSubState.Caption = "Waiting"
            ElseIf (Value = 3) Then
                lblSubState.Caption = "Running"
            Else
                lblSubState.Caption = "Unidentified"
            End If
        End If
    End If
End Sub
```

・接続先切り替え処理

StateChanged イベントで IPLink サーバとの接続状態を取得します。もし、接続状態が False であった場合、接続切り替えを行います。このサンプルでは、接続状態に変化があった場合、タイマーを使用して、インターバルを設けて接続の切り替え処理を行っています。これは、メインとサブ両方に接続ができない場合、頻繁に切り替え処理が発生することを防ぐためです。

タイマーイベント内では、接続状態を一度全て解除し、もう 1 つの NetworkParam を設定し直し、再度接続処理を行います。

●VB.Net

```
Private Sub IPLink1_StateChanged(ByVal sender As Object,
    ByVal e As AxIPLINKLib5._DClientIPLinkEvents_StateChangedEvent)
        Handles IPLink1.StateChanged

    '/// IPLink Status ////////////////////////////////////////
    lblIPLinkState.Text = CStr(e.state)

    '/// Redundancy Status Change ///////////////////////////////
    If (e.state = True) Then
        Timer1.Enabled = False
    Else
        Timer1.Interval = fvTimerInterval
        Timer1.Enabled = True
    End If
End Sub
```

```
Private Sub Timer1_Tick(ByVal Sender As Object, ByVal e As System.EventArgs) Handles Timer1.Tick
    Dim vResults As Object = Nothing
    lblMainState.Text = "No State"
    lblSubState.Text = "No State"

    IPLink1.Online = False
    IPLink1.CtlUpdate = False
    If (IPLink1.NetworkParam = fcNetworkParam1) Then
        IPLink1.NetworkParam = fcNetworkParam2
    Else
        IPLink1.NetworkParam = fcNetworkParam1
    End If
    IPLink1.Online = True
    IPLink1.CtlUpdate = True
End Sub
```

●VB6.0

```
Private Sub IPLink1_StateChanged(ByVal State As Boolean)

    '/// IPLink Status ////////////////////////////////////////
    lblIPLinkState.Caption = State

    '/// Redundancy Status Change ////////////////////////////////
    If (State = True) Then
        Timer1.Enabled = False
    Else
        Timer1.Interval = fvTimerInterval
        Timer1.Enabled = True
    End If
End Sub
```

```
Private Sub Timer1_Timer()
    Dim vResults As Variant
    lblMainState.Caption = "No State"
    lblSubState.Caption = "No State"

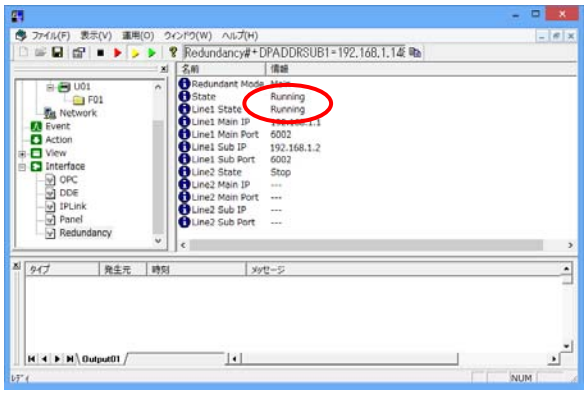
    IPLink1.Online = False
    IPLink1.Update = False
    If (IPLink1.NetworkParam = fcNetworkParam1) Then
        IPLink1.NetworkParam = fcNetworkParam2
    Else
        IPLink1.NetworkParam = fcNetworkParam1
    End If
    IPLink1.Online = True
    IPLink1.Update = True
End Sub
```

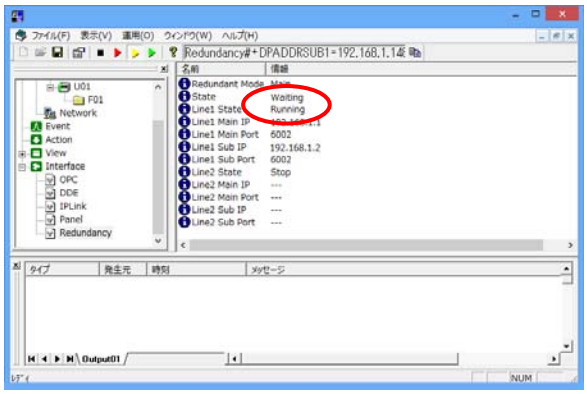


本サンプルでは接続状態に変化があった場合、タイマーを使用して、インターバルを設けて接続の切り替え処理を行っています。これは、メインとサブ両方に接続ができない場合、頻繁に切り替え処理が発生することを防ぐためです。インターバルの間隔は、変数「fvTimerInterval」で設定されており、値を変更することで、間隔を変えることができます。

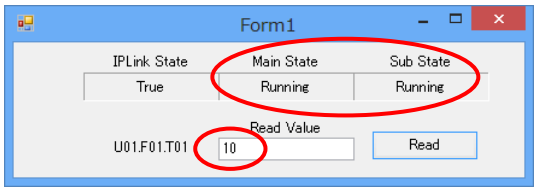
動作確認

メイン機アプリケーション、サブ機アプリケーションの順にオンライン化した後に、メイン機アプリケーションの Redundancy インターフェースの情報を表示させ、State と Line1 State が共に Running になっていることを確認して下さい。また、この際、後からオンライン化した方が待機状態になるため、サブ機側は State が Waiting になります。

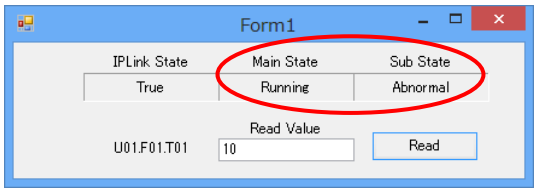


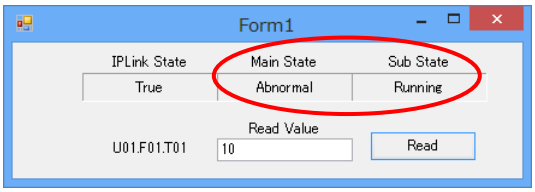


Visual Basic アプリケーションを実行し、Main と Sub の State が共に Running になることを確認してください。合わせて、Read ボタンを押すと、タグの値を取得できることを確認して下さい。



メイン機稼働中にサブ機がオフラインの場合、Sub State が Abnormal になることを確認して下さい。また、サブ機稼働中にメイン機がオフラインの場合、Main State が Abnormal になることを確認して下さい。





5章　リファレンス

5－1　機能一覧

メソッド・プロパティ	
■接続処理	
NetworkParam	IPLink サーバとの接続パラメータ。
Online	IPLink サーバと接続/切断を行う。
Update (CtlUpdate)	クライアントキャッシュの自動更新を開始/終了する。
■タグ登録・解除	
AddTag	タグを登録する。
AddTags	複数のタグを一括で登録する。
ActiveTag	タグのアクティブ状態を変更する。
ActiveTags	複数のタグのアクティブ状態を一括で変更する。
DelTag	タグの登録を解除する。
DelTags	複数のタグの登録を一括で解除する。
■通信	
ReadVal	タグの値を読み込む。
ReadVals	複数のタグの値を一括で読み込む。
WriteVal	タグに値を書き込む。
WriteVals	複数のタグに値を一括で書き込む。
■その他	
AccessPath	タグの共通パス名。
State (CtlState)	IPLink サーバとの接続状態（0-NG 1-OK）
ErrMessage	最後に発生したエラーのメッセージ内容。
イベント	
ValueStateChanged	値及び値品質が変化したときに発生する。
ValueChanged	値が変化したときに発生する。
ValueBitOn	ビットが On したときに発生する。
ValueBitOff	ビットが Off したときに発生する。
StateChanged	IPLink サーバの生存状態が変化したときに発生する。



「Update」と「State」プロパティは、VB.Net では予約語となっているため、「Ctlxxxx」という名前に置き換わります。

5－2　メソッド・プロパティ

NetworkParam プロパティ

IPLink サーバとの接続パラメータ。

設定

“IP アドレス,ポート番号,更新周期(ms),通信タイムアウト(秒),接続タイムアウト(秒)”
例）AxIPLink1.NetworkParam = “192.168.1.1,3000,1000,5,10”

解説

設定値	解説
IP アドレス	サーバの IP アドレスです。省略すると、起動した PC のデフォルトの IP アドレスが使用されます。
ポート番号	サーバのポート番号です。
更新周期(ms)	Update プロパティにより自動更新される周期です。省略すると 1000ms（1 秒）になります。
通信タイムアウト(秒)	通信時のタイムアウト時間です。省略すると 5 秒になります。
接続タイムアウト(秒)	接続時のタイムアウト時間です。省略すると 4 秒になります。

NetworkParam プロパティはオンラインにする前に設定しておく必要があります。接続先を変更する場合には、いったんオフラインにした後に NetworkParam の設定を変更して、再度オンラインにして下さい。尚、サーバとクライアントが 1 台の PC 上で実行される構成の場合、「IP アドレス」は指定する必要はありません。IP アドレスを省略することで、他の PC へ環境を移行する場合などに IP アドレス設定変更の手間を省略できます。反対に、クライアント側のプログラムをリモートの PC 上で稼動させる場合には、サーバの IP アドレスを明示する必要があります。

関連項目

Online プロパティ Update プロパティ

Online プロパティ

IPLInk サーバとのオンラインモードを切り替えます。

設定

AxIPLink1.Online = True

設定値	解説
False	オフライン。
True	オンライン。

解説

オンラインにすると NetworkParam プロパティに従い、IPLink サーバとの通信を開始します。

関連項目

NetworkParam プロパティ

Update (.Net の場合：CtlUpdate) プロパティ

	クライアントキャッシュの自動更新を開始/終了する。
設定	AxIPLink1.Update = True
解説	Update プロパティを True に設定すると、定期的にサーバから値を取得し、クライアントキャッシュを更新します。更新周期は NetworkParam プロパティで指定します。クライアントキャッシュからの値の取得は高速に処理されます。キャッシュから値を取得するには、RealVal/ReadVals メソッドを使用し、DataSouce 引数に「0-クライアントキャッシュ値」を指定します。自動更新によって値が変化すると、ValueChanged 等のイベントが発生します。尚、自動更新は Active なタグに対してのみ行われます。
関連項目	ReadVal メソッド　NetworkParam プロパティ　AddTag メソッド ActiveTag メソッド ValueChanged イベント

AddTag メソッド

	タグを登録する。
構文	bResult = AxIPLink1.AddTag(TagPath, Active)

引数	解説
bResult	False - 失敗。True - 成功。
TagPath	タグパスです。 例）U01.F01.T01
Active	False – アクティブでない。True – アクティブ。

解説	ReadVal/WriteVal メソッドなどで通信を行う前に、AddTag メソッドでタグを登録する必要があります。Active 引数に True を指定すると、サーバは PLC と定期的に通信を行い、サーバキャッシュを PLC の最新値で自動的に常に更新します。これにより、RealVal/ReadVals メソッドによって値を取得する際、DataSouce 引数に「1-サーバキャッシュ値」を指定する事によって、サーバキャッシュから値を取得することができます。Active 引数に False を指定した場合、サーバはキャッシュ自動更新のための定期的な PLC との通信を行いません。例えば、読み込み通信が必要ない場合、あるいは読み込み頻度が低い場合などは、Active 引数に False を指定することによって、サーバ・PLC 間の通信負荷を減らすことができます。Active 引数が False のタグに対して値を取得したい場合には、ReadVal/ReadVals メソッドの DataSouce 引数で「2-デバイス値」を指定して下さい。
エラー処理	bResult は、オフライン時には必ず成功になります。オンライン時、サーバへの登録が無事確立した場合のみ True になります。もし、オンライン時にタグの登録に失敗した場合、次回の通信時に自動的に登録のリトライが行われるため、AddTag を再度呼び出す必要はありません。
関連項目	ReadVal/ReadVals メソッド　WriteVal/WriteVals メソッド

AddTags メソッド

	複数のタグを一括で登録する。
構文	bResult = AxIPLink1.AddTags(Count, TagPath, Actives, Results)

引数	解説
bResult	False - 失敗。True - 成功。
Count	登録を行うタグの個数です。
TagPaths	タグパスを配列で指定します。
Actives	アクティブ状態を配列で指定します。
Results	タグごとの結果が返ってきます。bResult が True の時のみ有効です。

解説	AddTags メソッドは一度に複数のタグの登録ができます。従って、複数のタグを登録したい場合には、AddTag メソッドを複数回呼び出すよりも AddTags メソッドを呼び出した方が高速に処理されます。Results は、オフラインの場合は必ず成功になります。
エラー処理	Results は、オフライン時には必ず成功になります。オンライン時は、サーバへの登録が無事確立したときだけ、True になります。オンライン時にタグの登録に失敗した場合、次回の通信時に自動的に登録のリトライが行われるため、AddTags を再度呼び出す必要はありません。
関連項目	AddTag メソッド

ActiveTag メソッド

	タグのアクティブ状態を変更する。
構文	bResult = IPLink1.ActiveTag(TagPath, Active)

引数	解説
bResult	False - 失敗。True - 成功。
TagPath	タグパスです。 例）U01.F01.T01
Active	False – アクティブでない。True – アクティブ。

解説	AddTag メソッドで登録したタグのアクティブ状態を変更します。
エラー処理	bResult は、オフライン時には必ず成功になります。オンライン時は、サーバへ変更通知が無事確立したときだけ True になります。オンライン時にタグの登録に失敗した場合は次回の通信時に自動的にリトライが行われるため、ActiveTag を再度呼び出す必要はありません。
関連項目	AddTag メソッド

ActiveTags メソッド

	複数のタグのアクティブ状態を一括で変更する。
構文	bResult = AxIPLink1.ActiveTags(Count, TagPath, Actives, Results) 引数 / 解説 bResult: False - 失敗。True - 成功。 Count: 登録を行うタグの個数です。 TagPaths: タグパスを配列で指定します。 Actives: アクティブ状態を配列で指定します。 Results: タグごとの結果が返ってきます。bResult が True の時のみ有効です。
解説	ActiveTags メソッドは一度に複数のタグのアクティブ状態を変更することができます。従って、ActiveTag メソッドを複数回呼び出すよりも高速に処理が行われます。
エラー処理	Results は、オフライン時には必ず成功になります。オンライン時は、サーバへ変更通知が無事確立したときだけ、True になります。オンライン時にタグの登録に失敗した場合は次回の通信時に自動的にリトライが行われるため、ActiveTags を再度呼び出す必要はありません。
関連項目	ActiveTag メソッド

DelTag メソッド

	タグの登録を解除する。
構文	bResult = AxIPLink1.DelTag(TagPath, Active) 引数 / 解説 bResult: False - 失敗。True - 成功。 TagPath: タグパスです。 例）U01.F01.T01
解説	AddTag メソッドで登録したタグを解除します。
関連項目	AddTag メソッド

DelTags メソッド

	複数のタグの登録を一括で解除する。
構文	bResult = AxIPLink1. DelTags(Count, TagPath) 引数 / 解説 bResult: False - 失敗。True - 成功。 Count: 登録を行うタグの個数です。 TagPaths: タグパスを配列で指定します。
解説	DelTags メソッドは一度にタグの解除ができるため、DelTag メソッドを複数回呼び出すよりも高速に処理が行われます。
関連項目	DelTag メソッド

ReadVal メソッド

	タグの値を読み込みます。
構文	bResult = AxIPLink1.ReadVal(TagPath, Value, DataSource) 引数 / 解説 bResult: False - 失敗。True - 成功。 TagPath: タグパスです。 例）U01.F01.T01 Value: 読み込んだ値です。bResult が True の時のみ有効です。 DataSource: 0 – クライアントキャッシュ値。1 – サーバキャッシュ値。 2 – デバイス値。
解説	ReadVal メソッドで値を読み込む前に、あらかじめ AddTag メソッドでタグを登録しておく必要があります。 DataSource「0」のとき、クライアントキャッシュから値を取得します。 DataSource「1」のとき、サーバキャッシュから値を取得します。 DataSource「2」のとき、PLC デバイスの値を取得します。 詳しくは「クライアントキャッシュとサーバキャッシュ」を参照して下さい。
関連項目	AddTag メソッド　Active メソッド

ReadVals メソッド

複数のタグの値を一括で読み込む。

構文

bResult = AxIPLink1. ReadVals(Count, TagPaths, Values, Results, DataSource)

引数	解説
bResult	False - 失敗。True - 成功。
Count	読み込みを行うタグの個数です。
TagPaths	タグパスを配列で指定します。
Values	読み込んだ値が配列で返ってきます。bResult が True の時のみ有効です。
Results	タグごと値の品質が返ってきます。bResult が True の時のみ有効です。
DataSource	0 – クライアントキャッシュ値。1 – サーバキャッシュ値。 2 – デバイス値。

例）

```
Dim TagPaths(2) As String
Dim Values As Object = Noting     //VB6.0 の場合は Variant 型を指定
Dim Results As Object = Noting    //VB6.0 の場合は Variant 型を指定
TagPaths(0) = "U01.F01.T01"
TagPaths(1) = "U01.F01.T02"
If AxIPLink1.ReadVals(2, TagPaths, Values, Results, 1) Then
    For i = 0 To 1
        If Results(i) Then
            txtValue(i).Text = Values(i)
        Else
            txtValue(i).Text = "Error!"
        End If
    Next i
Else
    For i = 0 To 1
        txtValue(i).Text = "Error!"
    Next i
End If
```

解説

ReadVals は一度に複数タグの値を一括で読み込みます。従って、複数のタグの値を読み込みたい場面では、ReadVal メソッドを複数回呼び出すよりも高速に処理が行われます。ReadVals メソッドで値を読み込む前に、あらかじめ AddTag メソッドでタグを登録しておく必要があります。ReadVals の呼び出しに成功すると、返値に True が設定されます。この値が True の時のみ、Values/Results が有効になります。Values には取得したタグ値が、Results にはタグごとの値の品質（通信が成功したかどうか）が格納され、Result(x)が True の時、Values(x)に有効な値が格納されています。DataSource に関する詳細は ReadVal メソッドを参照してください。

関連項目

AddTag メソッド ReadVal メソッド

WriteVal メソッド

	タグに値を書き込む。
構文	bResult = AxIPLink1. WriteVal(TagPath, Value) 引数 / 解説 bResult / False - 失敗。True - 成功。 TagPath / タグパスです。 例）U01.F01.T01 Value / 書き込む値です。
解説	WriteVal メソッドで値を書き込む前に、あらかじめ AddTag メソッドでタグを登録しておく必要があります。
関連項目	AddTag メソッド

WriteVals メソッド

複数のタグに値を一括で書き込む。

構文

bResult = AxIPLink1. WriteVals(Count, TagPaths, Values, Results)

引数	解説
bResult	False - 失敗。True - 成功。
Count	書き込みを行うタグの個数です。
TagPaths	タグパスを配列で指定します。
Values	書き込む値をバリアント型配列で指定します。
Results	タグごとの書き込みの結果が返ってきます。bResult が True の時のみ有効です。

例）

```
Dim TagPaths(2) As String
Dim Values(2) As Object          //VB6.0 の場合は Variant 型を指定
Dim Results As Object            //VB6.0 の場合は Variant 型を指定
TagPaths(0) = "U01.F01.T01"
TagPaths(1) = "U01.F01.T02"
Values(0) = 123
Values(1) = 456
If AxIPLink1.WriteVals(2, TagPaths, Values, Results) Then
    For i = 0 To 1
        If Results(i) Then
            txtResult(i).Text = "Success!"
        Else
            txtResult(i).Text = "Error!"
        End If
    Next i
Else
    For i = 0 To 1
        txtResult (i).Text = "Error!"
    Next i
End If
```

解説

WriteVals メソッドは複数のタグに対して一度に値を書き込むことができます。従って、複数のタグ値に書き込みを行いたい場合、WriteVal メソッドを複数回呼び出すよりも、高速に処理が行われます。WriteVals メソッドで値を書き込む前に、あらかじめ AddTag メソッドでタグを登録しておく必要があります。WriteVals メソッドの呼び出しに成功すると、返値に True が返ります。Results は返値が True の時のみ有効で、タグ毎の通信の成否が格納されます。

関連項目

AddTag メソッド WriteVal メソッド

AccessPath プロパティ

	タグの共通パス名。
設定	例）AxIPLink1. AccessPath = “U01.F01.”
解説	AccessPath プロパティはタグ名の共通文字部分を指定することにより、下記のメソッドでタグパスの文字列の前半を省略できます。 ・AccessPath が有効なメソッド AddTag/AddTags ActiveTag/ActiveTags DelTag/DelTags ReadVal/ReadVals WriteVal/WriteVals 例えば AccessPath プロパティに“U01.F01.”を設定した場合、従来 AddTag(“U01.F01.T01”, xxx)と呼び出していた記述を、AddTag(“T01”, xxx)に省略することができます。 例） AxIPLink1.AccessPath = “U01.F01.” AxIPLink1.AddTag “T01”, TRUE AxIPLink1.ReadVal “T01”, a AxIPLink1.WriteVal “T01”, a AxIPLink1.DelTag “T01” ※内部的には AccessPath + “T01” すなわち“U01.F01.T01”で処理されています。 また、動的タグ書式を利用し、“U01.F01.#”とすることにより、PLC のアドレスをダイレクトに指定できます。 例） AxIPLink1.AccessPath = “U01.F01.#” AxIPLink1.AddTag “D0000”, TRUE ... ※内部的には AccessPath + “T01” すなわち“U01.F01.#D0000”で処理されています。
関 連 項目	AddTag/AddTags/ActiveTag/ActiveTags/DelTag/DelTags/ReadVal/ReadVals/WriteVal/WriteVals メソッド

State (.Net の場合：CtlState) プロパティ

	IPLink サーバとの接続状態。
取得	例）dState = AxIPLink1. State State　解説 State　0 – 未接続　1 – 接続中
解説	State プロパティが 1 の時、サーバと接続が行われている状態を表します。Online プロパティに True を設定したとしても、実際に接続が確立されていない場合、State プロパティは False になります。State プロパティが変化すると SateChanged イベントが発生します。
関連項目	StateChanged イベント

ErrMessage プロパティ

	最後に発生したエラーのメッセージ内容。
取得	例）strMessage = AxIPLink1. ErrMessage

解説	直近に発生したエラーメッセージが格納されています。各メソッドが失敗した直後に参照して下さい。
	例）
	If Not AxIPLink1.ReadVal("U01.F01.T01", varValue, 0) Then
	　txtError.Text = "Fail to read : " + IPLink1.ErrMessage
	End If

5－3　イベント

ValueStateChanged イベント

	値及び値品質が変化したときに発生する。
構文	●VB.Net Private Sub AxIPLink1_ValueStateChanged(sender As Object, e As AxIPLINKLib5._DClientIPLinkEvents_ValueStateChangedEvent) Handles AxIPLink1.ValueStateChanged 引数 / 解説 e.tagPath / タグパスです。 例）U01.F01.T01 e.value / 値です。 e.state / 値の状態です。異常時 False になります。 ●VB6.0 Private Sub IPLink1_ValueStateChanged (ByVal TagPath As String, ByVal Value As Variant, ByVal State As Boolean) 引数 / 解説 TagPath / タグパスです。 例）U01.F01.T01 Value / 値です。 State / 値の状態です。異常時 False になります。
解説	値及び値の状態が変化した時に発生するイベントです。値の状態は、正常に PLC と通信ができている場合は True です。PLC に異常が発生した場合、False になります。PLC に異常が発生した場合、Value には最後に正常な通信が行われた時のタグ値が格納されています。
関連項目	ValueChange イベント

ValueChanged イベント

	値が変化したときに発生する。
構文	●VB.Net Private Sub AxIPLink1_ValueChanged(sender As Object, e As AxIPLINKLib5._DClientIPLinkEvents_ValueChangedEvent) Handles AxIPLink1.ValueChanged 引数 / 解説 e.tagPath / タグパスです。 例）U01.F01.T01 e.value / 値です。 ●VB6.0 Private Sub IPLink1_ValueChanged (ByVal TagPath As String, ByVal Value As Variant) 引数 / 解説 TagPath / タグパスです。 例）U01.F01.T01 Value / 値です。
解説	値が変化した時に発生するイベントです。Value に格納される値は、常に通信が成功した時の値であることが保証されています。
関連項目	ValueStateChanged イベント

ValueBitOn/ValueBitOff イベント

ビットが On/Off したときに発生する。

構文

●VB.Net

```
Private Sub AxIPLink1_ValueBitOn(sender As Object,
  e As AxIPLINKLib5._DClientIPLinkEvents_ValueBitOnEvent)
    Handles AxIPLink1.ValueBitOn
```

引数	解説
e.tagPath	タグパスです。 例）U01.F01.T01
e.value	値です。

●VB6.0

```
Private Sub IPLink1_ValueBitOn
          (ByVal TagPath As String, ByVal Value As Variant)
```

引数	解説
TagPath	タグパスです。 例）U01.F01.T01
Value	値です。

解説

ビットが On/Off した時に発生するイベントです。Value に格納される値は、常に通信が成功した時の値であることが保証されています。なお、タグが数値型であった場合は、0（ゼロ）を Off として認識し、それ以外の値は On と解釈します。文字列の場合は、いったん数値に変換後、数値型と同様に扱われます。

関連項目

ValueChanged イベント

StateChanged イベント

	IPLink サーバの接続状態が変化したときに発生する。
構文	●VB.Net Private Sub AxIPLink1_StateChanged(sender As Object, e As AxIPLINKLib5._DClientIPLinkEvents_StateChangedEvent) Handles AxIPLink1.StateChanged 引数 / 解説 e.state / False – 未接続　True – 接続中 ●VB6.0 Private Sub IPLink1_StateChanged(ByVal State As Boolean) 引数 / 解説 State / False – 未接続　True – 接続中
解説	サーバとの接続状態が変化したときに発生するイベントです。
関連項目	State プロパティ

「IPLink インターフェース詳細マニュアル」Copyright © 2010 Roboticsware, Inc. All rights reserved.
2010 年 6 月 1 日第 発行　株式会社ロボティクスウェア
〒331-0811 埼玉県さいたま市北区吉野町 1-11-1　TEL:048-782-9861 FAX:048-782-9862

■本書は著作権上の保護を受けています。本書の一部または全部について、株式会社ロボティクスウェアの
文章での許諾を得ずに、いかなる方法においても無断で複写・複製することは禁じられています。
■記載されている会社名、製品名、ブランド名は、対応する法人または個人の（登録）商標です。
■このソフトウェアの仕様及び本書に記載されている事項は、将来予告無しに変更することがあります。