Roboticsware



マルチ PLC 通信 ActiveX

FA-Engine5 for VisualBasic

ユーザーズマニュアル

RWMN-5-21-016

ようこそ **5**
ご挨拶 5
マニュアルの読み方 6
１章　ご利用の前に **7**
概要 7
ランタイムライセンスの配布条件 7
ランタイムライセンスの配布方法 8
開発環境について 8
コンパイルについて 9
FA-Engine を Visual Basic .Net に組み込む 10
FA-Engine を Visual Basic 6.0 に組み込む 11
アプリケーション構成 12
仮想デバイスとは 13
２章　はじめての接続 **14**
２－１　はじめてのアプリケーション作成 14
準備 14
ステップ１（タグ設定） 14
ステップ２（プログラミング） 16
２－２　エラーの取得 21
エラープロパティによる取得 21
エラーイベントによる取得 21
ログファイルによる取得 22
３章　機能解説 **23**
重要インターフェース 23
一般的なプログラムの処理フロー 24
タグの登録 25
タグ設定ファイルの形式 25
接続処理 26
値の読み込み方法 26
自動更新の特徴 28
タグパス 29
書き込み後のキャッシュ更新について 30
システムプロパティ 30
イベント 31
４章　実践編 **32**
４－１　モニタリング（SAMPLE01） 32
概要 32
設定 32
プログラム 33
解説 34
４－２　データの読み込み（SAMPLE02） 35
概要 35
設定 35
プログラム 36

解説 37
４－３　データの書き込み（SAMPLE03） 38
概要 38
設定 38
プログラム 39
解説 41
４－４　大量なデータの扱い方（SAMPLE04） 42
概要 42
設定 42
プログラム 43
5章　リファレンス 45
５－１　一覧 45
５－２　メソッド・プロパティ 47
Connect / DisConnect メソッド 47
StartUpdate / StopUpdate メソッド 47
ReadVal メソッド 48
ReadVals メソッド 49
WriteVal メソッド 50
WriteVals メソッド 51
GetVal メソッド 52
SetVal メソッド 52
ReadRefresh メソッド 53
WriteRefresh メソッド 53
AliveTest メソッド 54
GetUnitState メソッド 54
AccessPath プロパティ 55
AddUnit メソッド 56
DelUnit メソッド 56
AddFolder メソッド 57
DelFolder メソッド 57
AddTag メソッド 58
DelTag メソッド 58
ClearAll メソッド 59
LoadFile / SaveFile メソッド 59
GetUnitCount メソッド 60
GetUnitName メソッド 60
GetUnitParam メソッド 60
GetFolderCount メソッド 61
GetFolderName メソッド 61
GetFolderParam メソッド 62
GetTagCount メソッド 62
GetTagName メソッド 63
GetTagParam メソッド 63
GetTagAddress メソッド 63
ErrCode プロパティ 64
ErrMessage プロパティ 64
エラー関連プロパティ 65
WriteErrorLog メソッド 65

AutoConnect / AutoUpdate プロパティ 66
AutoLoadFile / AutoLoadFileName プロパティ 66
StartAccessTimeLog メソッド 67
StopAccessTimeLog メソッド 67
ExecuteTagEditer メソッド 67
５－３　イベント 68
ValueStateChanged イベント 68
ValueChanged イベント 69
ValueBitOn/ValueBitOff イベント 70
ValueOverLimit イベント 71
ValueRapidChanged イベント 72
ValueInitialize イベント 73
ValueRefresh イベント 74
StateChanged イベント 75
ErrorOccured イベント 76
EndAccessTimeLog イベント 76
５－４　VER3 との比較 77
５－５　VER4 との比較 79
５－６　エラーコード 79

ようこそ

ご挨拶

このたびは、Roboticsware® FA シリーズをご購入いただきありがとうございます。

FA シリーズは 1996 年発売の「FA-Engine/FA-Touch Ver1.0」から数々の進化を遂げ、現在 Ver5 に至っております。FA シリーズは常に PLC との通信を中心として発展してきたソフトウェアであり、特に通信の高速性と安定性について細心の注意をはらい開発を進めてきました。

また、FA シリーズはファクトリーオートメーション開発者の発想を第一に考え、必要な機能をできるだけシンプルにまとめるよう様々な工夫が凝らされています。Ver5 は Ver4 からの安定性を引継ぎつつ、Ver1.0 当初からかわらぬ「シンプル＆高性能」のコンセプトをおしすすめています。

ぜひ、あなたの個性あふれる新しいファクトリーオートメーションの世界を作り上げてください。

Roboticsware 一同

マニュアルの読み方

FA-Engine のご利用に際しては、以下のマニュアル構成の中から必要なマニュアルを選択してお読み下さい。

●基本となるマニュアル構成は以下のとおりです。



■インストールガイド

FA シリーズ共通のインストールマニュアルです。製品をインストールする際にお読み下さい。



■FA-Engine ユーザーズマニュアル

本書です。FA-Engine の利用について記述されたマニュアルです。



■タグマニュアル

FA シリーズにおける重要な概念である「タグ」について解説されています。特に初めて FA シリーズを導入される場合、当該マニュアルによってタグの概念をある程度理解されることをお奨めします。



■接続マニュアル

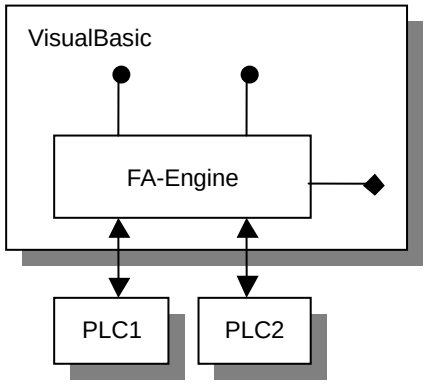
PLC との接続に関する、機種別の詳細解説書です。PLC と接続を行う際にお読み下さい。

1章　ご利用の前に

概要

FA-Engine は、VisualBasic などの開発環境で PLC のデータを扱うためのライブラリです。

FA-Engine は ActiveX 形式のライブラリであり、VisualBasic などの開発環境に組み込んで利用します。作成するアプリケーション以外に通信用の特別なアプリケーションを実行する必要が無く、ライブラリ自体が通信を行います。ReadVal/WriteVal などのメソッドや ValueChanged イベントを用いて、アプリケーションからの PLC のデバイス値の読み込み、書き込みが可能です。



ランタイムライセンスの配布条件

ActiveX 通信ライブラリを使用して作成された実行形式のアプリケーションは、自由に配布することができます（但し、不特定の第三者への配布は別途許諾が必要です)。本製品についての再配布時の条件は以下のとおりです。

１．配布方法
再配布条件に基づき、「FA-Engine5」ランタイム用インストーラを自由に配布することができます。ランタイム用インストーラ以外からのファイルの配布はできません。ランタイム用インストーラは CD-ROM「¥Runtime¥FAEngine」フォルダ内にあります。

２．著作権表示
本製品を配布する際、ロボティクスウェアの著作権表示をしていただく必要があります。著作権表示は以下のとおりに表示して下さい。

●アプリケーション、マニュアル、またはこれに類する物への表示方法
本製品は、株式会社ロボティクスウェアの著作物である「FA-Engine Ver5」を再配布条件に基づき使用しています。

●プログラム使用許諾契約書、またはこれに類する物への表示方法
本製品は、株式会社ロボティクスウェアの著作物である「FA-Engine Ver5」を再配布条件に基づき使用しています。株式会社ロボティクスウェアの著作物はバックアップの目的以外では一切の複写を許可しません。

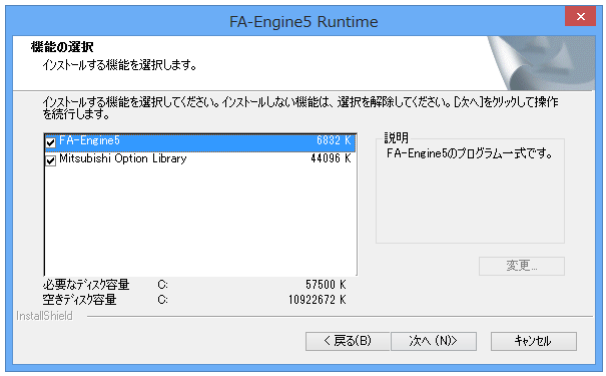
３．配布時のサポートについて
再配布先のサポートは行いません。サポートは、再配布元であるお客様にて行って下さい。

ランタイムライセンスの配布方法

ランタイムに必要なファイルのインストールには、CD-ROM の「¥Runtime¥FAEngine」にあるセットアッププログラムを実行します。このランタイム用インストーラは配布条件に基づき、自由にコピーして作成したアプリケーションと共に配布することができます。

ランタイムのインストール途中で「カスタム」を選択すると、インストール内容の詳細な選択が可能です。三菱非公開プロトコルによる通信を必要としない場合には、「Mitsubishi Option Library」のチェックを外すことによってより小さい容量でセットアップが行えます。





開発環境について

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。

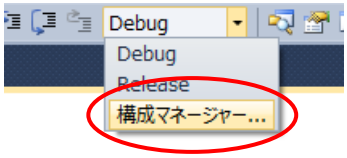
・VisualBasic 6.0
・VisualBasic .net（VisualStudio2003/2005/2008/2010)

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

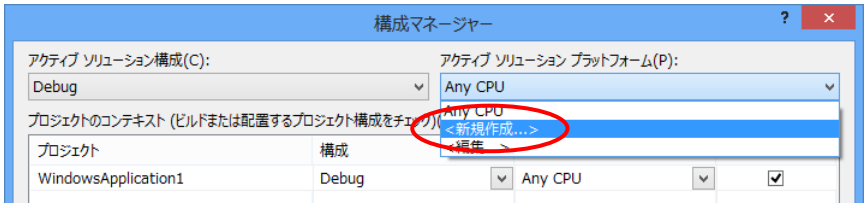
FA-Engine の OCX は 32Bit（x86）構成になっています。そのためアプリケーションのコンパイルを行う場合、32Bit（x86）指定でコンパイルを行う必要があります。以下の手順で構成を変更できます。

1. 構成マネージャーを選択する

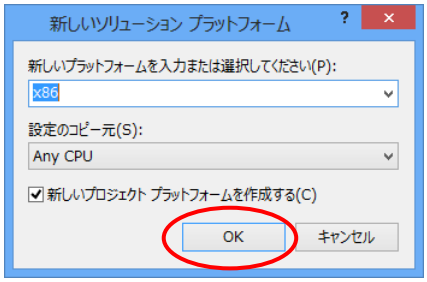


構成マネージャーが表示されない場合、[ツール] - [オプション] - [プロジェクトおよびソリューション] - [全般] を開き、[ビルド構成の詳細を表示]にチェックを入れてください。

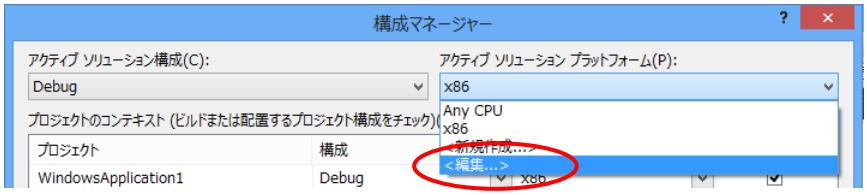
2. 下記の設定箇所から新規作成を選択する



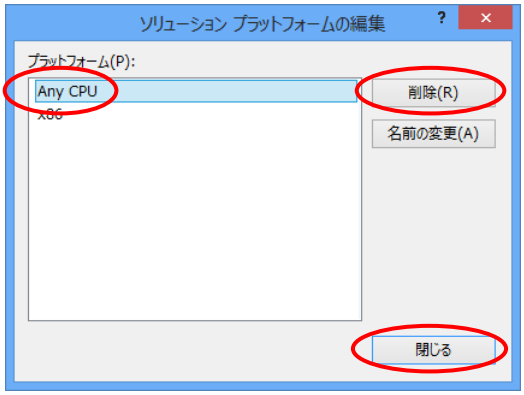
3. X86 を選択し OK を押下する



4. 編集を選択する



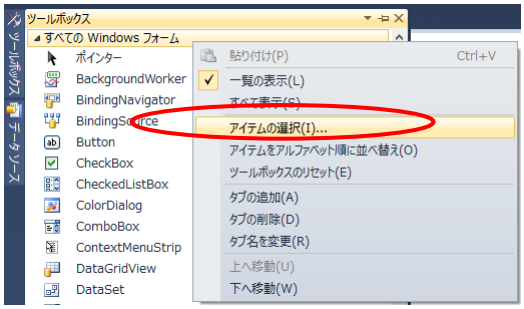
5. 「Any CPU」を選択して削除ボタンを押して設定を削除し、ダイアログを閉じる



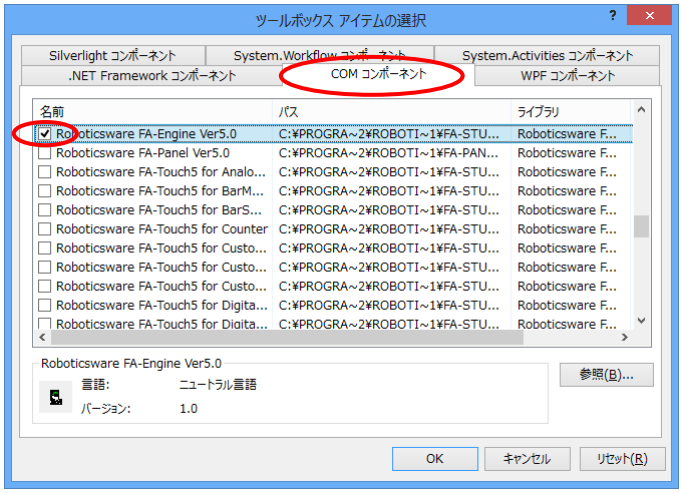
FA-Engine を Visual Basic .Net に組み込む

FA-Engine を利用してプログラムを作成する際には、VisualBasic のコンポーネント設定によって事前に登録しておく必要があります。VisualBasic2010 を例とした登録手順は下記のとおりです。

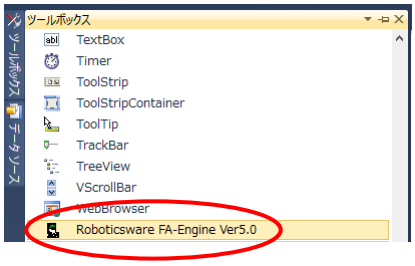
1. ツールボックスで右クリックし、メニュー［アイテムの選択］を選択する



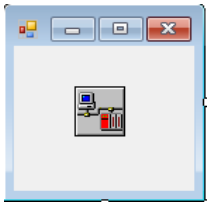
2. 「Roboticsware FA-Engine Ver5」にチェックを付けて、「OK」をクリックする



3. ツールボックスに以下のアイコンが追加されたのを確認する



利用する際には、アイコンを選択してフォームへ貼り付けてください。



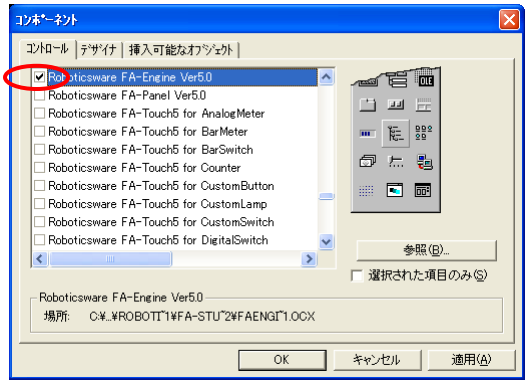
FA-Engine を Visual Basic 6.0 に組み込む

FA-Engine を利用してプログラムを作成する際には、VisualBasic のコンポーネント設定によって事前に登録しておく必要があります。VisualBasic6.0 の登録手順は下記のとおりです。

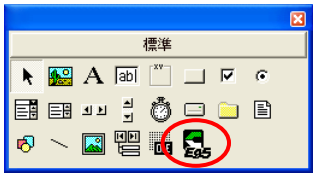
1. 「プロジェクト」メニューの「コンポーネント」を選択する



2. 「Roboticsware FA-Engine Ver5」にチェックを付けて、「OK」をクリックする



3. ツールボックスに以下のアイコンが追加されたのを確認する



利用する際には、アイコンを選択してフォームへ貼り付けてください。



アプリケーション構成

FA-Engine は、以下の 2 つのファイルで構成されています。

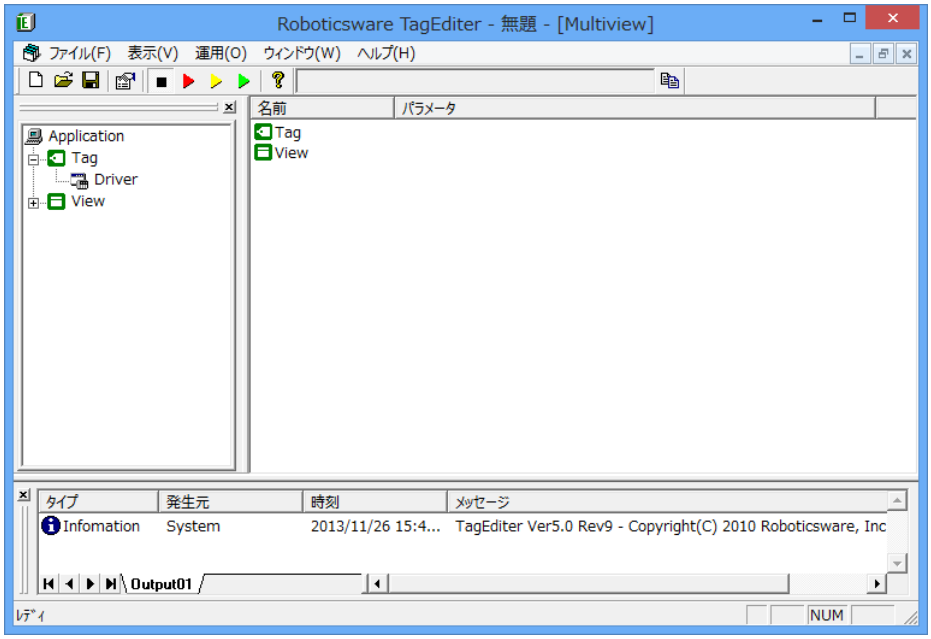
1．ActiveX 通信ライブラリ（FAEngine5.ocx）

PLC との通信を行う ActiveX コントロールです。コントロール単体で、対応する全ての PLC との通信が可能になります。尚、以降の記述において「FA-Engine」と表記されている場合には、当該 ActiveX コントロールのことを指すものとします。



2．タグエディタ（FAEngine5.exe）

「タグ」を編集するためのエディタです。タグエディタで編集した設定ファイルは、最終的に「FA-Engine」で読み込まれて使用されます。



仮想デバイスとは

現在 FA シリーズは 100 機種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、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。

２章　はじめての接続

２－１　はじめてのアプリケーション作成

準備

ここでは、PLC との通信を行うための入門的な手順について解説します。ここで紹介する手順は、大きく分けて以下の２ステップです。

ステップ１	タグの設定	通信対象の PLC デバイスをタグとして登録します。
ステップ２	プログラミング	VisualBasic からの読み込み、書き込み、モニタリング（動的更新）のプログラム作成例を紹介します。

この手順を進めるにあたっては、下記の環境を準備して下さい。
・FA-Engine
・VisualBasic

ステップ１（タグ設定）



タグ設定

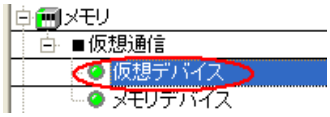
ここでは、仮想デバイスを用いて PLC をシミュレートするため、実機 PLC は手元に必要ありません。PLC が手元にある環境であれば、実際の PLC のタグを設定してもかまいません。

１．タグエディタ（FA-Engine.exe）を起動

２．左側のツリーの「Driver」を右クリックし「ユニット追加」を選択



３．「メモリ」の「仮想通信」「仮想デバイス」を選択し、「OK」を押す



４．「U01」を右クリックし「フォルダ追加」を選択する



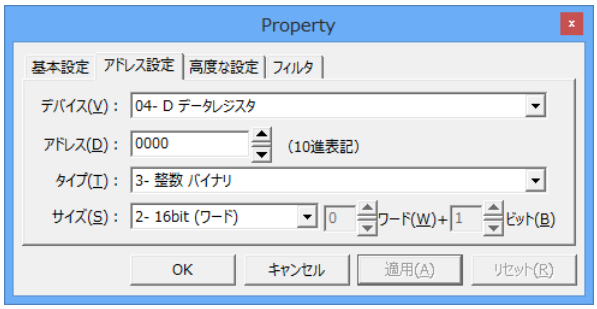
5. 「F01」を右クリックし「タグ追加」を選択



6. 左側のリストから「T01」を右クリックし、「プロパティ」を選択



7. 「アドレス設定」タブから「04 - D データレジスタ」を選択し、「OK」を押す
   下記の例では、D0000（ワードレジスタ）を数値タグとして設定しています。



8. 設定を保存する
   タグエディタの「ファイル」メニューから「名前を付けて保存」を選択して設定を保存してください。
   ここでは「c:¥sample.txt」というファイル名で保存します。

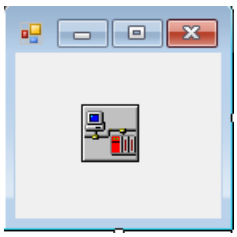
ステップ２（プログラミング）

開発環境の準備

１．VisualBasic を起動する

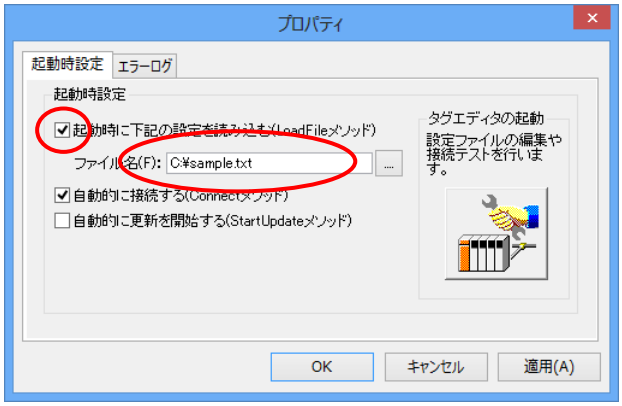
２．「FA-Engine」をフォームへ貼り付ける

１章で解説されている手順「FA-Engine を Visual Basic 6.0 に組み込む」もしくは「FA-Engine を Visual Basic .Net に組み込む」に従って、「FA-Engine」をフォームへ貼り付けてください。



３．プロパティ設定

「FA-Engine」を右クリックし、メニューから「プロパティ」を選択後、下記のように設定します。

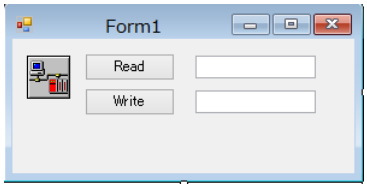


この設定により、「ステップ１」の作業で作成した設定ファイルが自動的に読み込まれるようになります。

ReadVal/WriteVal を使った実装

1．コーディング

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て、下記のコードを記述してください。



●VB.Net

読み込みボタン

```
Private Sub Button1_Click(ByVal sender As System.Object,
                          ByVal e As System.EventArgs) Handles Button1.Click
    If Not AxEng1.ReadVal("U01.F01.T01", a) Then
        TextBox1.Text = "Error!"
    Else
        TextBox1.Text = a
    End If
```

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 AxEng1.WriteVal("U01.F01.T01", TextBox2.Text) Then
        TextBox2.Text = "Error!"
    End If
End SubEnd Sub
```

●VB6.0

読み込みボタン

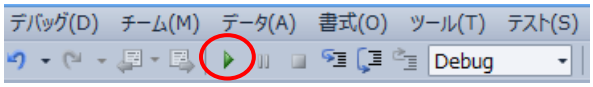
```
Private Sub Command1_Click()
    If Not Eng1.ReadVal("U01.F01.T01", a) Then
        Text1.Text = "Error!"
    Else
        Text1.Text = a
    End If
End Sub
```

書き込みボタン

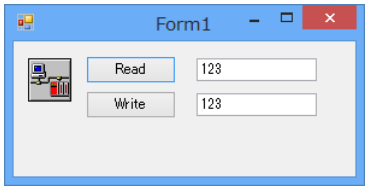
```
Private Sub Command2_Click()
    If Not Eng1.WriteVal("U01.F01.T01", Text2.Text) Then
        Text2.Text = "Error!"
    End If
End Sub
```

2．プログラム実行

VisualBasic の開始ボタンを押して、プログラムを開始してください。



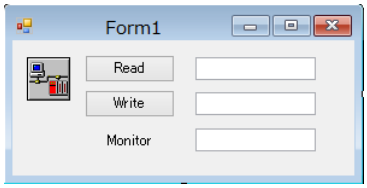
画面から任意の数値を入力し、書き込みボタン 及び 読み込みボタンの操作を行います。操作によって表示内容が変化することを確認してください。



リアルタイムモニタ

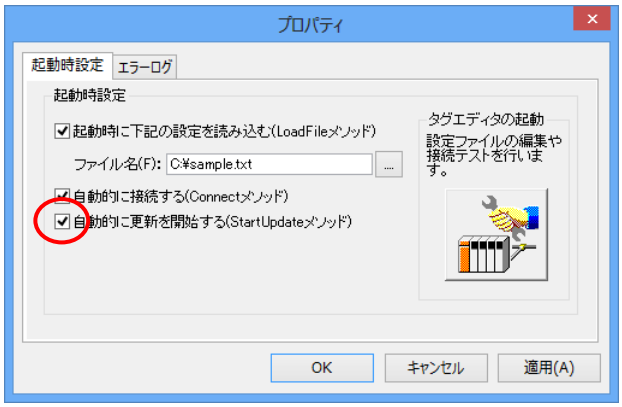
1．コーディング

先ほど作成したフォームの下部に、モニター用のテキストコントロールを貼り付けてください。



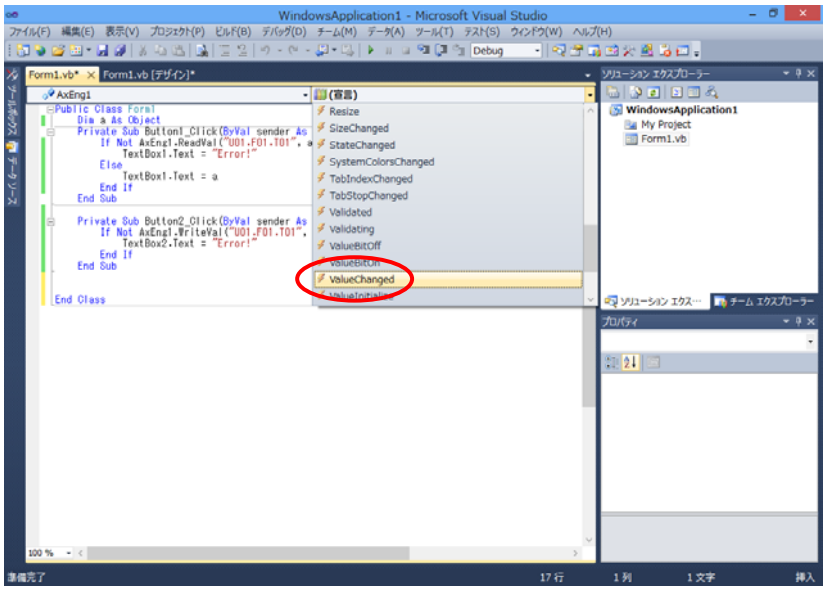
2．自動更新プロパティの設定

FA-Engine のプロパティで、「自動的に更新を開始する」にチェックを入れてください。



3．ValueChanged イベントの追加

FA-Engine  をダブルクリックし、「ValueChanged」イベントを追加してください。



「ValueChanged」イベントを追加後、下記のコードを記述します。

●VB.Net

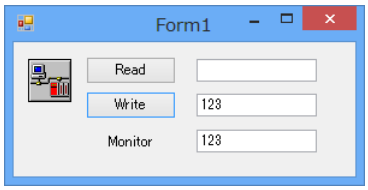
```
Private Sub AxEng1_ValueChanged(ByVal sender As Object,
    ByVal e As AxFAENGINELib5._DFAEngineEvents_ValueChangedEvent)
            Handles AxEng1.ValueChanged
    Select Case e.tagPath
        Case "U01.F01.T01"
            TextBox3.Text = e.value
    End Select
End Sub
```

●VB6.0

```
Private Sub Eng1_ValueChanged(ByVal TagPath As String, ByVal Value As Variant)
    Select Case TagPath
        Case "U01.F01.T01"
            Text3.Text = Value
    End Select
End Sub
```

4．プログラム実行

VisualBasic の開始ボタンを押して、プログラムを開始して下さい。



プログラムを開始す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。画面から任意の数値を書き込み、テキストコントロール（Text3）の表示内容が書き込みに応じて動的に変化することを確認して下さい。

２－２　エラーの取得

通信処理においてエラーが発生した場合、下記の３つの方法によってエラーの内容を知ることができます（更なる詳細については３章を参照して下さい）。

エラープロパティによる取得

ReadVal/WriteVal などのメソッド呼び出しが失敗すると、下記のプロパティにエラーコードとメッセージが格納されます。このプロパティの参照によって発生したエラーのエラーコードとメッセージを知ることができます。

Eng1.ErrCode
Eng1.ErrMessage

エラーイベントによる取得

エラーが発生した時には、エラーイベント（ErrorOccured）が呼び出されます。
イベントに渡されたエラーコードとメッセージによってエラーの内容を知ることができます。

●VB.Net

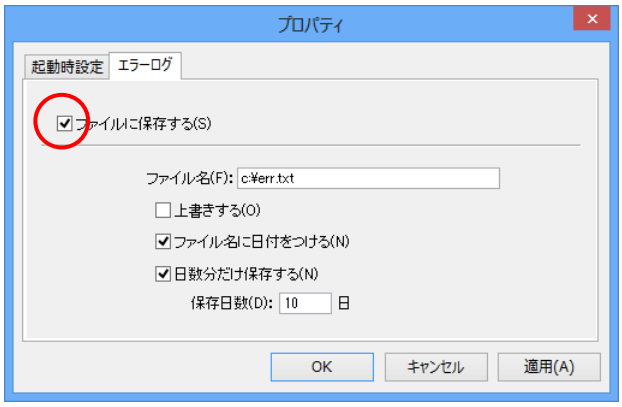
```
Private Sub AxEng1_ErrorOccured(ByVal sender As Object,
    ByVal e As AxFAENGINELib5._DFAEngineEvents_ErrorOccuredEvent)
        Handles AxEng1.ErrorOccured
    txtCode.text = e.code
    txtError.text = e.message
End Sub
```

●VB6.0

```
Private Sub Eng1_ErrorOccured(ByVal Code As Long, ByVal Message As String)
    txtCode.text = Code
    txtError.text = Message
End Sub
```

ログファイルによる取得

設定によって、エラー発生の履歴をエラーログファイルに保存することができます。
この設定は FAEngine5.ocx のプロパティで設定します。



3章　機能解説

重要インターフェース

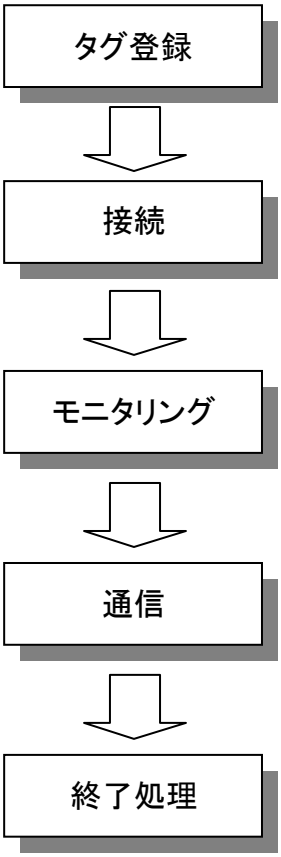
下表は、重要なインターフェースのみの抜粋です。その他のインターフェースについては5章を参照ください。

最重要：これらの利用のみでも通信可能　　○のないもの：場合により重要
重要　：重要につき要チェック　　表にないもの：必須では無いが利用すると便利

メソッド・プロパティ		最重要	重要
■通信			
Connect/DisConnect	PLC と接続/切断する。		○
StartUpdate/StopUpdate	キャッシュの自動更新を開始/終了する。		○
ReadVal	値を読み込む。	○	○
ReadVals	複数のタグの値を読み込む。		○
WriteVal	値を書き込む。	○	○
WriteVals	複数のタグの値を書き込む。		○
GetVal	キャッシュからタグの値を読み込む。	○	○
■タグ登録・解除			
AddUnit	ユニット（PLC）を登録する。		
DelUnit	ユニット（PLC）の登録を解除する。		
AddFolder	フォルダを登録する。		
DelFolder	フォルダの登録を解除する。		
AddTag	タグを登録する。		
DelTag	タグの登録を解除する。		
ClearAll	すべてのユニット、フォルダ、タグを削除する。		
LoadFile/SaveFile	タグ設定ファイルを読み込み/書き込みする。		
■エラー処理			
ErrCode	直近に発生した最新のエラーコード。	○	○
ErrMessage	直近に発生した最新のエラーメッセージ。	○	○
イベント			
ValueStateChanged	値及び値品質が変化したとき発生。		○
ValueChanged	値が変化したとき発生。	○	○
ValueBitOn	ビットが On したとき発生。		
ValueBitOff	ビットが Off したとき発生。		
ValueRefresh	自動更新のための定周期通信が完了するごとに発生。		
StateChanged	ユニットのデバイスダウンの状態が変化した時に発生。		○
ErrorOccured	エラー発生時に発生するイベント。	○	○

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。

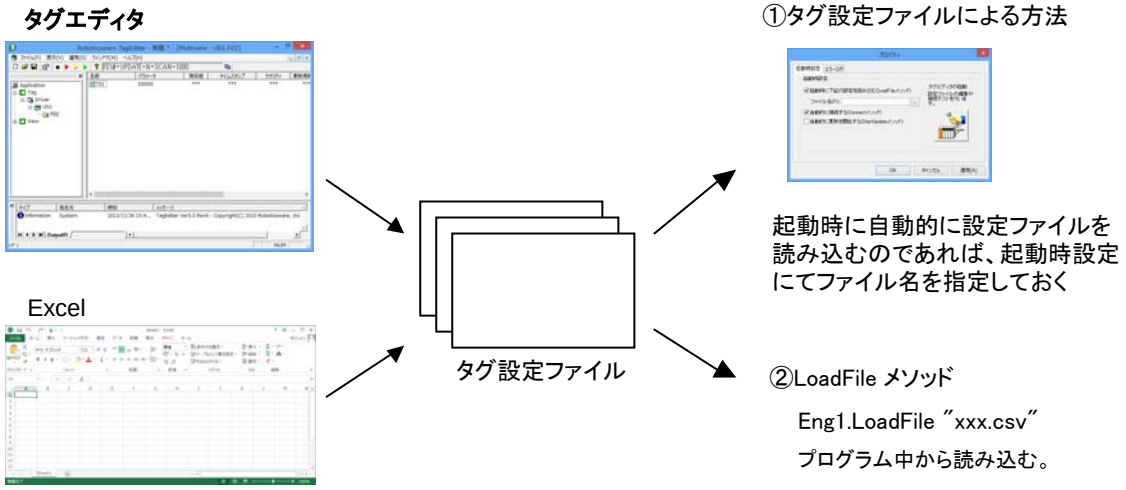


LoadFile AddTag	タグの登録をします。タグの設定はタグエディタで行います。尚、AddTag メソッドによってプログラム中から直接タグを追加することもできます。
Connect	PLC と接続します。接続後は ReadVal/WriteVal などの通信メソッドが有効になります。
StartUpdate ValueChanged	StartUpdate メソッドを呼び出すと、自動更新を開始し、値が変化した時に ValueChanged イベントが発生します。
ReadVal WriteVal	メソッド呼び出しによる読み書きです。ボタン操作に応じたオンデマンドの読み書きなどはこの方法で行います。
StopUpdate DisConnect	フォームを閉じると終了処理が自動的に行われるため、通常は意識する必要はありません。

タグの登録

タグの作成方法には以下の 2 つの方法があります。

①タグ設定ファイルによる方法
②メソッドによる方法

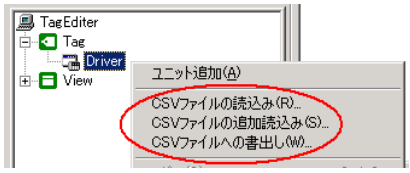


①タグ設定ファイルによる登録
あらかじめタグエディタでタグ設定ファイルを作成しておき、VisualBasic のプログラム内から読み込む方法です。この方法による読みこみでは「LoadFile」メソッドを使用します。例えば、タグ設定ファイルを複数用意しておき、任意のタイミングでプログラム内からタグ設定を切り替える事などが可能です。メソッドによる読み込みの他、アプリケーションの起動時に自動的に設定ファイルを読み込むように指定することもできます。この設定は FA-Engine の「起動時設定」プロパティページの設定で行うことができます。

②メソッドによる個別登録
AddTag/DelTag 等のメソッドを利用して、プログラム内からタグの登録を自由に行うことができます。この方法はタグ設定ファイルを読み込んだ後でも、タグの追加・削除を任意に行うことができます。尚、GetTagName/GetTagParam 等のメソッドを利用する事で、設定されているタグの情報をプログラムから参照することもできます。

タグ設定ファイルの形式

タグディタによって保存される設定ファイルは XML 形式のテキストファイルです。設定ファイルにはタグ設定情報の他、タグエディタ独自の設定内容も含まれています。設定ファイルの形式は、「FA-Studio」などの全ての FA シリーズアプリケーションで互換性があり、テキストエディタなどで直接編集することもできます。
尚、純粋にタグの情報だけを編集したい場合には、タグの設定内容を CSV ファイル形式で書き出し、あるいは読み込むための機能が用意されています。この機能によって書き出した CSV ファイルは、Excel などで直接編集して再度取り込む事などが可能であり、大量のタグを編集する際に便利な機能です。



接続処理

PLCとの接続処理はConnectメソッドによって行います。ReadVal/WriteValなどによるPLCとの通信は、Connectメソッドの呼び出し以降に行うことができます。尚、「FA-Engine」のプロパティの起動時設定で、「自動的に接続処理を行う」にチェックを入れておく事で、自動的に Connect メソッドが呼び出されるようになります。

■オープン処理の注意事項
Connect メソッドでは、通信のための初期処理が行われます。例えば RS-232C 接続であれば COM ポートをオープンし、Ethernet 接続であればソケットのオープン処理を行います。これらはユニットごとに処理されますので、例えば重複した RS-232C ポートが各ユニットから設定されていた場合には、有効になるのは１つのユニットだけです。この動作は、複数の「FA-Engine」コントロールを貼り付けて同様の構成で設定を行った場合にも同様の現象になります。

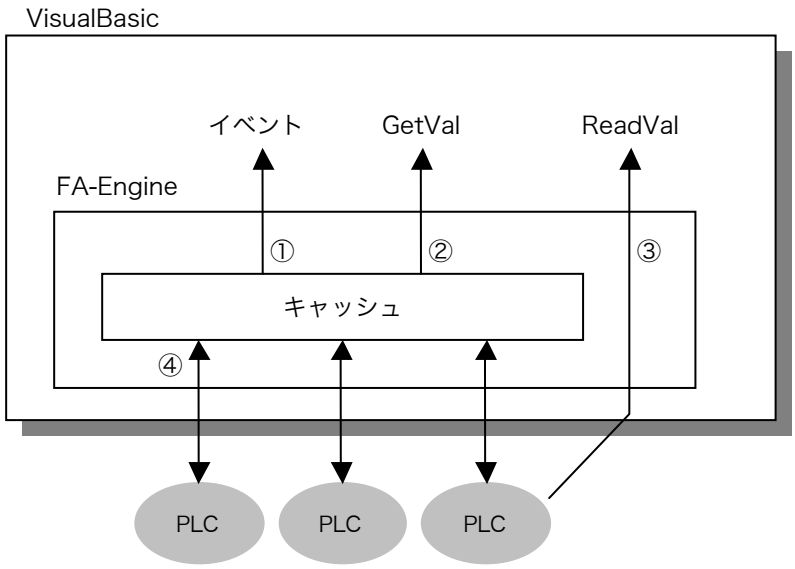
値の読み込み方法

データの読み込みには下記の３つの方法があります。

・イベントを利用　　ValueChanged イベントなど
・キャッシュを利用　　GetVal メソッド
・直接通信　　ReadVal メソッド

「FA-Engine」にはタグの値を一時的に保存するためのキャッシュが存在します。キャッシュの自動更新④はStartUpdate メソッドにより開始します。キャッシュには自動更新によって定期的に最新の値が書き込まれ、値が変化した時に①のように ValueChanged イベントなどが発生します。また、キャッシュの値は②GetVal メソッドで取得できます。この自動更新のための周期通信（ポーリング）は別タスクで行われるため、VisualBasic の動作を妨げず、モニタリング処理の実装などに最適です。なお、自動更新の周期はフォルダのプロパティで設定します。

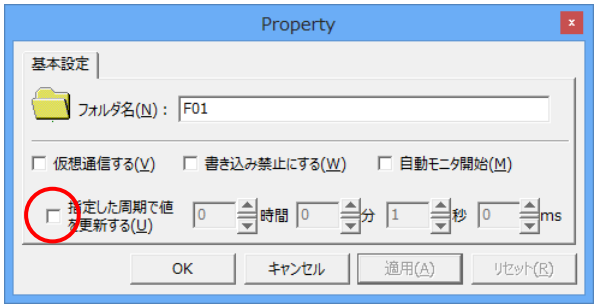
また、③の ReadVal メソッドはキャッシュを参照せず、メソッドが呼び出される度に PLC とのダイレクトな通信によってデバイス値を取得します。従って、応答速度は GetVal に比べ遅くなりますが、最新のデータを取得することが可能です。この方法は同期を伴う制御の実装や、バッチ的な書き込み処理の実装に向いています。



下記は各読込み方法の条件と特徴です。

通信方法	特徴	StartUpdate
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。自動更新のための通信は、VisualBasic とは異なるタスクで実行されるため、VisualBasic 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。	必須
GetVal （通常時、推奨）	GetVal によるキャッシュからの値所得は、処理速度が最も速いのが特徴です。VisualBasic 上で、迅速に値を所得したい場合は、この方法をお奨めします。	必須
ReadVal （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。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。	必要なし

これら３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、あるタグを StartUpdate により定期的に読み込みを行いながら ReadVal でさらに通信を行うという組み合わせは、最もパフォーマンスが低下する例です。周期通信＋ダイレクト通信になってしまい、必要のない周期通信の分、通信負荷を無駄に上げてしまいます。また、より高いパフォーマンスを得るためには、定周期通信が必要なタグとその他のタグのフォルダを分けて登録します。そして、周期通信が必要のないフォルダは「指定した周期で値を更新する」のチェックを外します。



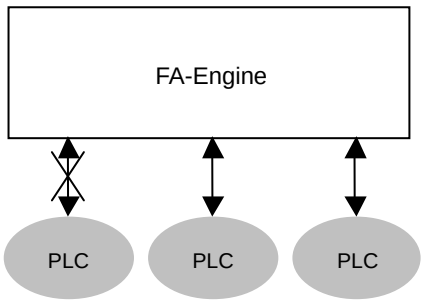
これらの通信方法を特に意識しない場合は、すべてのタグにおいて StartUpdate メソッドにより自動更新を行い、GetVal メソッド ＋ ValueChanged イベントの組み合わせで値を取得することをお奨めします。

自動更新の特徴

自動更新のための通信には、下記のような特徴があります。

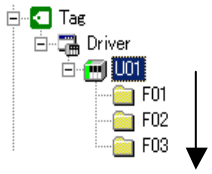
●StartUpdate を呼び出す事によって自動更新が開始します（※必要の無いときは呼び出さないで下さい）。

●PLC 毎の通信は独立しており、並列的に処理されるため、他の PLC との通信負荷に影響を及ぼしません。PLC との接続台数が増えても通信速度に変化はありません。



PLC 毎に通信が独立しているため、例えば 1 台の PLC が通信不能になっても他の PLC の通信に影響はない。

●通信はフォルダ単位に行われる。複数のフォルダが定義されている場合には、順番に通信が行われる。



フォルダ単位の通信が終了する毎に ValueRefresh イベントが呼び出されます。

●フォルダのプロパティで、「指定した周期で値を更新する」がチェックされているものだけが自動更新されます。



●自動更新の周期はフォルダのプロパティの更新周期によって設定します。



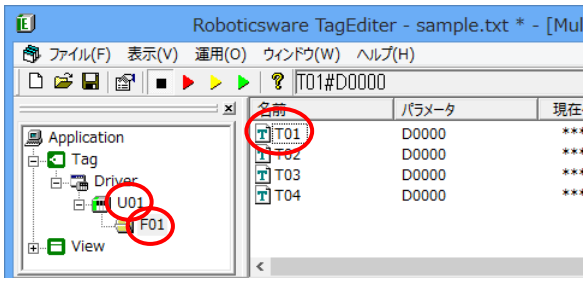
●通信に伴う各イベントを発生させます。例えば、ValueRefresh イベントはフォルダ単位の通信が終了する毎に呼び出されます

●GetVal メソッドでキャッシュの値を高速に取得することができます。

●通信エラーは「ErrorOccured」イベントでキャッチすることができます。

タグパス

タグは下記の３階層で設定されます
・ユニット　U01 など
・フォルダ　F01 など
・タグ　　T01 など



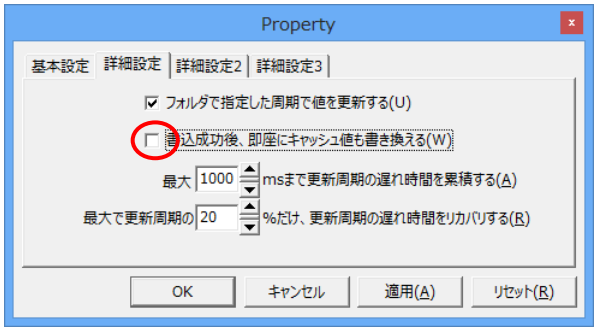
ReadVal/WriteVal/Value などのメソッドではタグパスを指定する必要があります。タグパスは下記のように３階層を「.」（ピリオド）でつなげた文字列です。

AxEng1.ReadVal “U01.F01.T01”, value
　　タグパス

また、AccessPath プロパティを用いることにより、一括してタグパスの前半文字列を省略することができます。

```
AxEng1Eng1.AccessPath = “U01.F01.”
AxEng1Eng1.ReadVal “T01”, a
AxEng1Eng1.ReadVal “T02”, b
AxEng1Eng1.ReadVal “T03”, c
AxEng1Eng1.ReadVal “T04”, d
```

書き込み後のキャッシュ更新について



タグの設定では、PLC に対して書き込みが成功した場合、即座に内部キャッシュの値を書き換えるかどうかを設定できます。

■チェックされている場合（デフォルト）
書き込み成功後、ValueChange 等のイベントが発生します。また、Value プロパティでの値の取得でも、書き込み後の値が渡されます。

■チェックされていない場合
次の PLC との読込通信が行われるまで、ValueChange 等のイベントは発生しません。また Value プロパティによる値も、次の PLC との読込通信によりキャッシュがリフレッシュされるまでは以前の値が返されます。

システムプロパティ

ユニット、フォルダ、タグはそれぞれ、下記のようなシステムプロパティを持っています。システムプロパティの参照ではオブジェクト名とプロパティ名を「!」でつなげます。

U01!Alive / U01!Virtual / U01!DriverMain など
F01!Alive / F01!Virtual / F01!UpdateInterval など
T01!ResetFilter など

上記のような参照によって、プログラムからシステムプロパティの値を設定/取得する事ができます。例えば「F01!UpdateInterval」はフォルダの更新周期を表し、下記のような構文で更新周期を変更することができます。

AxEng1Eng1.WriteVal “F01!UpdateInterval”, 1000
システムプロパティ

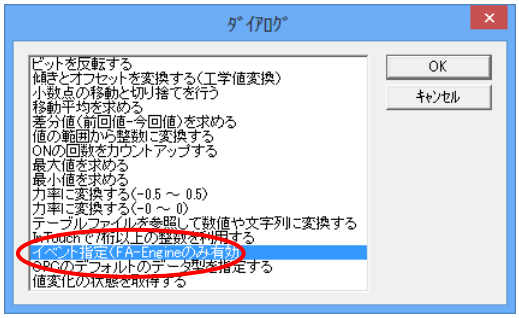
システムプロパティについての詳細は、「タグマニュアル」を参照ください。

イベント

StartUpdate の呼び出しによって、自動更新が開始し、値が変化した時に各イベントが発生するようになります。
デフォルトでは、下記のイベントが発生する仕様になっています。

イベント	デフォルト	設定の可否
ValueStateChanged	発生する	固定
ValueChanged	発生する	切替可
ValueBitOn	発生する	切替可
ValueBitOff	発生する	切替可
ValueOverLimit	発生しない	切替可
ValueRapidChanged	発生しない	切替可
ValueInitialize	発生する	切替可
ValueRefresh	発生する	固定
StateChanged	発生する	固定
ErrorOccured	発生する	固定
EndAccessTimeLog	発生する	固定

イベントを発生させる、させない、あるいは各イベントのフィルタのパラメータを変更する場合には、タグプロパティの「フィルタ」タブで「イベント指定」フィルタを追加し、各イベントの条件を設定して下さい。







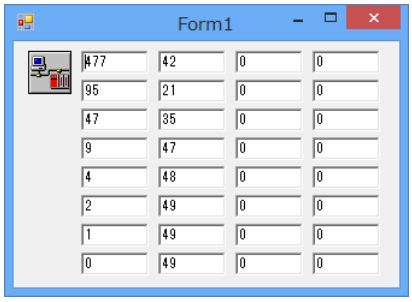
ValueChanged/ValueBitOn/ValueBitOff などはデフォルトで発生するようになっています。デフォルトの状態でＯＫであれば、「イベント指定」フィルタを追加する必要はありません。

４章　実践編

４－１　モニタリング（Sample01）

概要

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。



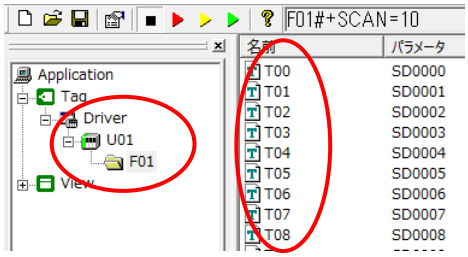
このサンプルは以下に格納されています。
¥¥sample¥faengine¥VB6～VB.Net2010¥Sample01

設定



タグ設定

タグ「T00」～「T31」を設定します。



プログラム

値変化イベントを記述します。

●VB.Net

```
Private Sub AxEng1_ValueStateChanged(ByVal sender As Object,
    ByVal e As AxFAENGINELib5._DFAEngineEvents_ValueStateChangedEvent)
        Handles AxEng1.ValueStateChanged
    If e.state Then
        '正常時
        txtVal(Integer.Parse(e.tagPath.Substring(e.tagPath.Length - 2))).Text =
            CStr(e.value)
    Else
    '異常時
        txtVal(Integer.Parse(e.tagPath.Substring(e.tagPath.Length - 2))).Text = "Error!"
    End If
```

●VB6.0

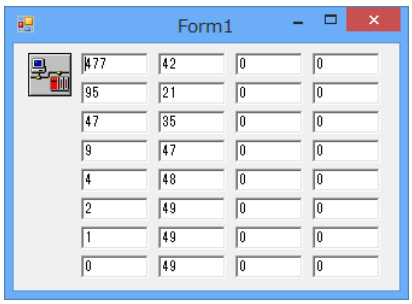
```
Private Sub Eng1_ValueStateChanged(ByVal TagPath As String,
    ByVal Value As Variant, ByVal State As Boolean)
    If State Then
        '正常時
        txtValue(CLng(Right(TagPath, 2))).Text = Value
    Else
        '異常時
        txtValue(CLng(Right(TagPath, 2))).Text = "Error!"
    End If
End Sub
```



更新周期はフォルダのプロパティで設定します。



解説

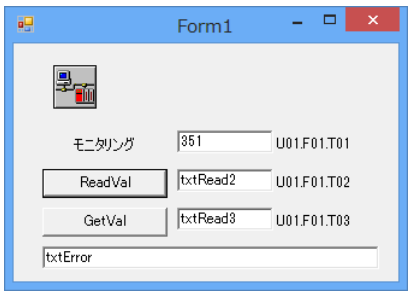


プログラムを実行すると、値が変化するたびに ValueStateChanged イベントが呼び出されます。PLC との通信ができない状態の場合、ValueStateChanged イベントの State 引数が FALSE になります。

４－２　データの読み込み（Sample02）

概要

３つの方法で値を読み込みするサンプルです。

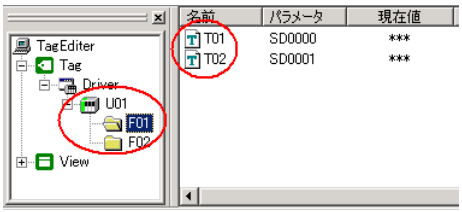


このサンプルは以下に格納されています。
¥¥sample¥faengine¥VB6～VB.Net2010¥Sample02

設定

タグ設定

タグ「U01.F01.T01」「U01.F01.T02」「U01.F02.T01」を設定します。



プログラム

読み込み処理を記述します。

●VB.Net

```
'読み込み１
Private Sub AxEng1_ValueChanged(ByVal sender As Object,
    ByVal e As AxFAENGINELib5._DFAEngineEvents_ValueChangedEvent)
        Handles AxEng1.ValueChanged
    Select Case e.tagPath
        Case "U01.F01.T01"
            txtRead1.Text = CStr(e.value)
    End Select
End Sub

'読み込み２
Private Sub btnRead2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles btnRead2.Click
    txtRead2.Text = CStr(AxEng1.GetVal("U01.F01.T02"))
End Sub

'読み込み３
Private Sub btnRead3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles btnRead3.Click
    Dim a As Object = Nothing
    If AxEng1.ReadVal("U01.F02.T01", a) Then
        txtRead3.Text = CStr(a)
    Else
        txtError.Text = CStr(AxEng1.ErrCode) & ":" & AxEng1.ErrMessage
    End If
End Sub
```

●VB6.0

```
'読み込み１
Private Sub Eng1_ValueChanged(ByVal TagPath As String, ByVal Value As Variant)
    Select Case TagPath
        Case "U01.F01.T01"
            txtRead1.Text = Value
    End Select
End Sub

'読み込み２
Private Sub btnRead2_Click()
    txtRead2.Text = Eng1.GetVal("U01.F01.T02")
End Sub

'読み込み３
Private Sub btnRead3_Click()
    If Eng1.ReadVal("U01.F02.T01", a) Then
        txtRead3.Text = a
    Else
        txtError.Text = Eng1.ErrCode + ":" + Eng1.ErrMessage
    End If
End Sub
```

解説

このサンプルのポイントは、自動更新が必要なタグとそれ以外のタグを、２つのフォルダで別々に管理している点です。モニタリング及び GetVal ではキャッシュの自動更新が必要ですが、ReadVal は自動更新が不要です。そこで、自動更新有りフォルダ「F01」及び自動更新無しフォルダ「F02」を作成し、それぞれにタグを分けて登録しています。結果として無駄な通信が発生しません。

自動更新有りフォルダ「F01」
U01.F01.T01 / U01.F01.T02



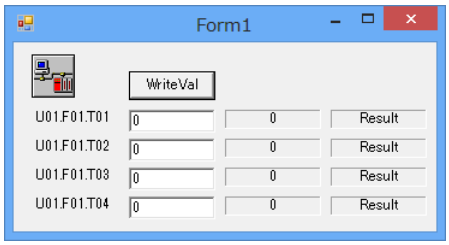
自動更新無しフォルダ「F02」
U01.F02.T01



４－３　データの書き込み（Sample03）

概要

値を書き込むサンプルです。

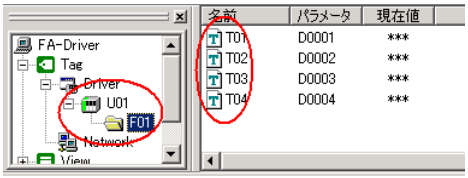


このサンプルは以下に格納されています。
¥¥sample¥faengine¥VB6～VB.NET2010¥Sample03

設定

タグ設定

タグ「T01」～「T04」を設定します。



プログラム

'書き込み処理を記述します。

●VB.Net

```
'書き込み
Private Sub btnWrite_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles btnWrite.Click
    Dim TagPaths(4) As String
    Dim Values(4) As Object    '必ずオブジェクトで宣言して下さい。
    Dim Results As Object = Nothing
    TagPaths(0) = "U01.F01.T01"
    TagPaths(1) = "U01.F01.T02"
    TagPaths(2) = "U01.F01.T03"
    TagPaths(3) = "U01.F01.T04"
    Values(0) = txtWrite(0).Text
    Values(1) = txtWrite(1).Text
    Values(2) = txtWrite(2).Text
    Values(3) = txtWrite(3).Text

    Dim i As Integer
    If AxEng1.WriteVals(4, TagPaths, Values, Results) Then
        For i = 0 To 3
            If CBool(Results(i)) Then
                lblResult(i).Text = "Success!"
            Else
                lblResult(i).Text = "Error!"
            End If
        Next i
    Else
        For i = 0 To 3
            lblResult(i).Text = "Error!"
        Next i
    End If
End Sub
```

●VB6.0

```
Private Sub btnWrite_Click()

    Dim TagPaths(4) As String
    Dim Values(4) As Variant
    Dim Results As Variant

    TagPaths(0) = "U01.F01.T01"
    TagPaths(1) = "U01.F01.T02"
    TagPaths(2) = "U01.F01.T03"
    TagPaths(3) = "U01.F01.T04"

    Values(0) = txtWrite(0).Text
    Values(1) = txtWrite(1).Text
    Values(2) = txtWrite(2).Text
    Values(3) = txtWrite(3).Text

    If Eng1.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
End Sub
```

解説

WriteVals は複数のタグに一括で書き込むことができます。複数の連続したアドレス（D0000 から D0100 など）に対して書き込みが必要な場合、WriteVal を複数回呼び出すよりも WriteVals でまとめて処理した方が、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、配列タグの利用によって高速かつ簡単に通信を行うことが可能です。

WriteVals そのものの成否は返値で返却され、タグ毎の成否は Results 引数にブール値で格納されます。Results 引数は WriteVals が成功したときのみ有効になります。
従って、上記サンプルコードでは、WriteVals の返値で WriteVals 自体の処理が成功したかどうかの判別を行い、WriteVals が成功したときのみ Results 引数をチェックしています。

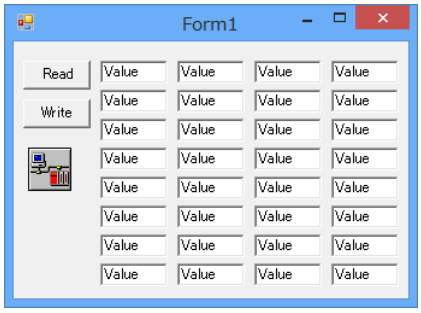


WriteVals での Values 引数は必ずオブジェクト型(VB6.0 は Variant 型)の配列で宣言して下さい（書き込む値をブール型、数値型、文字列型、ブール配列型、数値配列型、文字列配列型の複数の異なる型を混在して処理できるようにする為）。

４－４　大量なデータの扱い方（Sample04）

概要

配列タグを用いると大量のデータを１つのタグで通信できます。

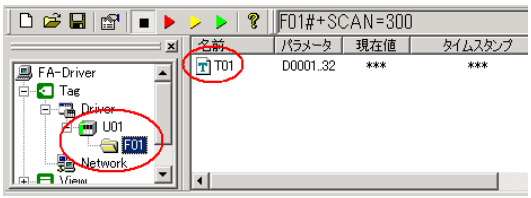


このサンプルは以下に格納されています。
¥¥sample¥faengine¥VB6～VB.Net2010¥Sample04

設定

タグ設定

配列タグ「T01」を設定します。
配列タグについては「タグマニュアル」を参照して下さい。



プログラム

読み込み及び書き込み処理を記述します。

●VB.Net

```
Private Sub btnRead_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles btnRead.Click
    Dim varValue As Object = Nothing
    Dim i As Integer
    If AxEng1.ReadVal("U01.F01.T01", varValue) Then
        For i = 0 To 31
            txtVal(i).Text = CStr(varValue(i))
        Next i
    Else
        For i = 0 To 31
            txtVal(i).Text = "Error!"
        Next i
    End If
End Sub

'書込み
Private Sub btnWrite_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles btnWrite.Click
    Dim varValue(32) As Double
    Dim i As Integer
    For i = 0 To 31
        varValue(i) = CDbl(txtVal(i).Text)
    Next i

    If Not AxEng1.WriteVal("U01.F01.T01", varValue) Then
        MsgBox("Fail to write. " & AxEng1.ErrMessage)
    End If
```

●VB6

```
'読込み
Private Sub btnRead_Click()
    Dim varValue As Variant
    If Eng1.ReadVal("U01.F01.T01", varValue) Then
        For i = 0 To 31
            txtValue(i).Text = varValue(i)
        Next i
    Else

    For i = 0 To 31
        txtValue(i).Text = "Error!"
    Next i
    End If
End Sub

'書込み
Private Sub btnWrite_Click()
    Dim varValue(32) As Double
    For i = 0 To 31
        varValue(i) = CDbl(txtValue(i).Text)
    Next i

    If Not Eng1.WriteVal("U01.F01.T01", varValue) Then
        MsgBox "Fail to write. " + Eng1.ErrMessage
    End If
End Sub
```

5章　リファレンス

5－1　一覧

メソッド・プロパティ	
■通信	
Connect / DisConnect	PLC との接続/切断を行う。
StartUpdate / StopUpdate	キャッシュの自動更新を開始/終了する。
ReadVal	値を読み込む。
ReadVals	複数のタグの値を読み込む。
WriteVal	値を書き込む。
WriteVals	複数のタグの値を書き込む。
GetVal	キャッシュからタグの値を読み込む。
SetVal	キャッシュへタグの値を書き込む。
ReadRefresh	フォルダ単位で読み込みを行い、キャッシュを更新する。
WriteRefresh	フォルダ単位でキャッシュ上の値を書き込む。
AliveTest	PLC との接続テストを行う。
GetUnitState	PLC の状態を取得する。
AccessPath	タグの共通パス名。
■タグ登録・解除	
AddUnit	ユニット（PLC）を登録する。
DelUnit	ユニット（PLC）の登録を解除する。
AddFolder	フォルダを登録する。
DelFolder	フォルダの登録を解除する。
AddTag	タグを登録する。
DelTag	タグの登録を解除する。
ClearAll	すべてのユニット、フォルダ、タグを削除する。
LoadFile/SaveFile	タグ設定ファイルの読み込み/書き込み。
■タグ参照	
GetUnitCount	ユニット数の取得。
GetUnitName	ユニット名の取得。
GetUnitParam	ユニット（PLC）のパラメータを取得。
GetFolderCount	フォルダ数の取得。
GetFolderName	フォルダ名の取得。
GetFolderParam	フォルダのパラメータを取得。
GetTagCount	タグ数の取得。
GetTagName	タグ名の取得。
GetTagParam	タグのパラメータを取得。
GetTagAddress	タグのアドレスを取得。

＜次頁につづく＞

＜前頁のつづき＞

メソッド・プロパティ	
■エラー処理	
ErrCode	直近に発生した最新のエラーコード。
ErrMessage	直近に発生した最新のエラーメッセージ。
ErrLog	True の時、エラーログを取得する。
ErrLogFileName	エラーログのファイル名。
ErrLogOverWrite	True の時、エラーログを上書きする。
ErrLogAddDateToFileName	True の時、エラーログファイル名に日付を付加する。
ErrLogSaveDays	True の時、一定期間外のエラーログファイルを自動削除する。
ErrLogSaveDaysTerm	エラーログファイルを自動削除する期間。
WriteErrorLog	エラーログにユーザーメッセージを書く。
■起動時設定	
AutoConnect / AutoUpdate	True の時、自動的に接続処理/更新処理を行う。
AutoLoadFile	True の時、タグ設定ファイルを自動的に読み込む。
AutoLoadFileName	自動的に読み込むタグ設定ファイル名。
■その他	
StartAccessTimeLog / StopAccessTimeLog	アクセスタイムログを開始/終了する。
ExecuteTagEditer	タグエディタを起動する。
イベント	
ValueStateChanged	値及び値品質が変化したとき発生。
ValueChanged	値が変化したとき発生。
ValueBitOn	ビットが On したとき発生。
ValueBitOff	ビットが Off したとき発生。
ValueOverLimit	上下限値を超えたとき発生。
ValueRapidChanged	値に急速な変化があったとき発生。
ValueInitialize	初回のみ発生。
ValueRefresh	自動更新のための通信が完了するごとに発生。
StateChanged	ユニットのデバイスダウンの状態が変化した時に発生。
ErrorOccured	エラーが発生時に発生するイベント。
EndAccessTimeLog	アクセスタイムログが終了したとき発生。



本項で紹介しているメソッド、プロパティ及びイベントの構文や例に記載されているコードは VB6 ベースのものになっています。VB.Net を使用する場合は、VB.Net のものに置き換えて使用してください。

5－2　メソッド・プロパティ

Connect / DisConnect メソッド

PLC との接続/切断を行います。

構文　　bResult = AxEng1.Connect([UnitName])

引数	解説
bResult	False - 失敗。True - 成功。
UnitName	接続/切断を行いたいユニット名を指定します。指定がない場合は、すべてのユニットの接続/切断処理を行います。

解説　　Connect メソッド呼び出し後、PLC と通信が行えます。DisConnect を呼び出すと、PLC とのコネクションを切断します。

StartUpdate / StopUpdate メソッド

キャッシュの自動更新を開始/終了します。

構文　　bResult = AxEng1.StartUpdate([UnitName])

引数	解説
bResult	False - 失敗。True - 成功。
UnitName	自動更新の開始/終了を行いたいユニット名を指定します。指定がない場合は、すべてのユニットの開始/終了処理を行います。

解説　　StartUpdate メソッドを呼び出すと、定期的に PLC から値を取得し、キャッシュを更新します。キャッシュ上の値は GetVal メソッドにより高速に取得できます。また、自動更新により値が変化した時 ValueChanged 等のイベントが発生します。更新周期はフォルダで指定します。また、自動更新はフォルダの「指定した周期で値を更新する」がチェックされているフォルダのみ行われます（自動更新は VisualBasic とは別のタスクで行われるため、VisualBasic の動きを止めません。）

関連項目　　ReadVal メソッド　ValueChanged イベント

ReadVal メソッド

値を読み込みます。

構文　　bResult = AxEng1.ReadVal(TagPath, Value)

引数	解説
bResult	False - 失敗。True - 成功。
TagPath	タグパス。 例）U01.F01.T01
Value	読み込んだ値。bResult が True の時のみ有効です。

解説　　PLC から値を読み込みます。メソッドが呼び出される度に通信が行われます。複数のタグの値を読み込みたい場合は ReadVals の方が高速に処理できます。

関連項目　　ReadVals メソッド

ReadVals メソッド

複数のタグの値を読み込みます。

構文　　bResult = AxEng1. ReadVals(Count, TagPaths, Values, Results)

引数	解説
bResult	False - 失敗。True - 成功。
Count	読み込みを行うタグの個数。
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。

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 AxEng1.ReadVals(2, TagPaths, Values, Results) 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　　複数のタグの値を読み込みたい場合、ReadVals は一度に読み込みが行われるため、ReadVal メソッドを複数回呼び出すよりも高速に処理されます。返値が True の時のみ、Values/Results が有効になります。Results ではタグごとの値の品質（通信が成功したかどうか）がわかります。Result(x)が True の時、Values(x)に有効な値が格納されています。

関連項目　　ReadVal メソッド

WriteVal メソッド

値を書き込みます。

構文　　bResult = AxEng1. WriteVal(TagPath, Value)

引数	解説
bResult	False - 失敗。True - 成功。
TagPath	タグパス。 例）U01.F01.T01
Value	書き込む値。

解説　　PLC に値を書き込みます。メソッドが呼び出される度に通信が行われます。複数のタグの値を書き込みたい場合は WriteVals の方が高速に処理できます。

関連項目　　WriteVals メソッド

WriteVals メソッド

複数のタグの値を書き込みます。

構文　　bResult = AxEng1. 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 = 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"
Values(0) = 123
Values(1) = 456
If AxEng1.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　　複数のタグの値を書き込みたい場合、WriteVals メソッドは一度に書き込みが行われるため、WriteVal メソッドを複数回呼び出すよりも、高速に処理されます。返値が True の時のみ、Results が有効になります。Results ではタグごとの通信の成否がわかります。

関連項目　　WriteVal メソッド

GetVal メソッド

キャッシュから値を取得します。

構文　　Value = AxEng1. GetVal(TagPath)

引数	解説
Value	タグの値。
TagPath	タグパス。 例）U01.F01.T01

解説　　キャッシュから値を高速に取得できます。キャッシュを定期的に更新するため StartUpdate メソッドを呼び出す必要があります。

関連項目　　StartUpdate メソッド

SetVal メソッド

キャッシュへ値を設定します。

構文　　AxEng1. SetVal(TagPath, Value)

引数	解説
Value	書き込む値。
TagPath	タグパス。 例）U01.F01.T01

解説　　キャッシュに値を設定します。主に、WriteRefresh で値を書き込む前に、あらかじめキャッシュに値を設定する際に利用されます。このとき StartUpdate メソッドは呼び出せません。

関連項目　　StartUpdate メソッド　WriteRefresh メソッド

ReadRefresh メソッド

フォルダ単位で読み込みを行い、キャッシュを更新します。

構文　　bResult = AxEng1.ReadRefresh(UnitName, FolderName)

引数	解説
bResult	False - 失敗。True - 成功。
UnitName	ユニット名。 例）U01
FolderName	フォルダ名。 例）F01

解説　　PLC から指定されたフォルダ内の全てのタグ値を読み込み、キャッシュをリフレッシュします。キャッシュ上の値は GetVal メソッドで高速に取得できます。StartUpdate による自動更新を行わず、独自のタイミングでキャッシュをリフレッシュしたい時に使用します。尚、ReadVals メソッドでも同等の処理の代替が可能です。

関連項目　　GetVal メソッド StartUpdate メソッド ReadVals メソッド

WriteRefresh メソッド

フォルダ単位でキャッシュ上の値を書き込みます。

構文　　bResult = AxEng1.WriteRefresh(UnitName, FolderName)

引数	解説
bResult	False - 失敗。True - 成功。
UnitName	ユニット名。 例）U01
FolderName	フォルダ名。 例）F01

解説　　キャッシュ上の値を PLC へ書き込みます。キャッシュへの値の設定には SetVal メソッドを使用します。StartUpdate による自動更新を併用した場合、キャッシュの値が常に更新されてしまうため、SetVal メソッドでの設定が無効になる場合があります。尚、WriteVals メソッドでも同等の処理の代替が可能です。

関連項目　　SetVal メソッド StartUpdate メソッド WriteVals メソッド

AliveTest メソッド

PLC との接続テストを行います。

構文　　bResult = AxEng1.AliveTest([UnitName], [FolderName])

引数	解説
bResult	False - 失敗。True - 成功。
UnitName	ユニット名。省略された場合は第 1 番目のユニットが対象となる。
FolderName	フォルダ名。省略された場合は第 1 番目のフォルダが対象となる。

解説　　PLC と実際に通信を行い、生存確認を行います。

GetUnitState メソッド

ユニットの状態（PLC の状態）を取得します。

取得　　dState = AxEng1. GetUnitState(UnitName)

引数	解説
dState	ユニット状態。1-Alive 2-Virtual 各値の OR
UnitName	ユニット名。　例）U01

解説　　dState は以下の様な組み合わせで 0～3 の値が格納されます。

・Alive=F/ Virtual =F ：0
・Alive=T/ Virtual =F ：1
・Alive=F/ Virtual =T ：2
・Alive=T/ Virtual =T ：3

例えば、(dState And 1)=True の時、PLC との通信は確立されています。(dState And 2)=True の時、仮想通信モードになっています。

関連項目　　StateChanged イベント

AccessPath プロパティ

タグの共通パス名です。

設定　　例）AxEng1. 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・プロパティ
ReadVal/ReadVals メソッド
WriteVal/WriteVals メソッド
Value プロパティ

例えば AccessPath プロパティに“U01.F01.”を設定した場合、従来 ReadVal(“U01.F01.T01”, xxx)と呼び出していたところを、ReadVal (“T01”, xxx)と省略できます。

例）
AxEng1.AccessPath = “U01.F01.”
AxEng1.ReadVal “T01”, a
AxEng1.WriteTag “T01”, a
※内部的には AccessPath + “T01” すなわち“U01.F01.T01”で処理されています。

関連項目　　ReadVal/ReadVals/WriteVal/WriteVals メソッド GetVal メソッド

AddUnit メソッド

ユニット（PLC）を登録します。

構文　　bResult = AxEng1.AddUnit(ParmStr)

引数	解説
bResult	False - 失敗。True - 成功。
ParmStr	“ユニット名#パラメータ文字列” 例）” U01#MIT.QQ.E:,8192,0.0.0.0,8192,UDP,0,3000,N,0,Y,Y,0”

解説　　AddUnit メソッドでタグの第 1 階層であるユニットをプログラム中から追加します。パラメータ文字列については「タグマニュアル」を参照ください。

関連項目　　DelUnit メソッド

DelUnit メソッド

ユニット（PLC）の登録を解除します。

構文　　bResult = AxEng1.DelUnit(UnitName)

引数	解説
bResult	False - 失敗。True - 成功。
UnitName	ユニット名。例）” U01”

解説　　DelUnit メソッドでタグの第 1 階層であるユニットをプログラム中から削除します。ユニット内に定義されているフォルダやタグも同時に削除されます。

関連項目　　AddUnit メソッド

AddFolder メソッド

フォルダを登録します。

構文　　bResult = AxEng1.AddFolder(UnitName , ParmStr)

引数	解説
bResult	False - 失敗。True - 成功。
UnitName	ユニット名。例）" U01"
ParmStr	“フォルダ名#パラメータ文字列” 例）"F01#00,FF"

解説　　AddFolder メソッドでタグの第２階層であるフォルダをプログラム中から追加します。パラメータ文字列については「タグマニュアル」を参照ください。

関連項目　　DelFolder メソッド

DelFolder メソッド

フォルダの登録を解除します。

構文　　bResult = AxEng1.DelFolder(UnitName, FolderName)

引数	解説
bResult	False - 失敗。True - 成功。
UnitName	ユニット名。例）" U01"
FolderName	フォルダ名。例）" F01"

解説　　DelUnit メソッドでタグの第２階層であるフォルダをプログラム中から削除します。フォルダ内に定義されているタグも同時に削除されます。

関連項目　　AddFolder メソッド

AddTag メソッド

タグを登録します。

構文　　bResult = AxEng1.AddTag(UnitName, FolderName, ParmStr)

引数	解説
bResult	False - 失敗。True - 成功。
UnitName	ユニット名。例）" U01"
FolderName	フォルダ名。例）" F01"
ParmStr	“タグ名#パラメータ文字列” 例）"T01#X000000"

解説　　AddTag メソッドでタグの第３階層であるタグをプログラム中から追加します。パラメータ文字列については「タグマニュアル」を参照ください。

関連項目　　DelTag メソッド

DelTag メソッド

タグの登録を解除します。

構文　　bResult = AxEng1.DelTag(UnitName, FolderName, TagName)

引数	解説
bResult	False - 失敗。True - 成功。
UnitName	ユニット名。例）" U01"
FolderName	フォルダ名。例）" F01"
TagName	タグ名。例）" T01"

解説　　DelUnit メソッドでタグの第３階層であるタグをプログラム中から削除します。

関連項目　　AddTag メソッド

ClearAll メソッド

すべてのユニット、フォルダ、タグを削除します。

構文　　bResult = AxEng1.ClearAll()

引数	解説
bResult	False - 失敗。True - 成功。

解説　　ユニット、フォルダ、タグをすべて削除し DisConnect メソッド及び StopUpdate メソッドを呼び出します。

関連項目　　DelUnit メソッド DelFolder メソッド DelTag メソッド DisConnect メソッド StopUpdate メソッド

LoadFile / SaveFile メソッド

タグ設定ファイルの読み込み/書き込みを行います。

構文　　bResult = AxEng1.LoadFile([FileName])

引数	解説
bResult	False - 失敗。True - 成功。
FileName	ファイル名。ファイル名を指定しなかった場合はファイルを指定するためのダイアログが表示されます。

解説　　タグ設定ファイルを読み込みます。LoadFile はアプリケーション形式（xml）、CSV 形式（タグ情報のみ）の両フォーマットとも読み込むことが可能です。SaveFile はタグ情報を CSV 形式ファイルで出力します。

関連項目　　AutoLoadFile / AutoLoadFileName プロパティ

GetUnitCount メソッド

ユニット数を取得します。

取得　　lCount = AxEng1. GetUnitCount

解説　　現在設定されているユニット数を返します。

GetUnitName メソッド

ユニット名を取得します。

取得　　strName = AxEng1. GetUnitName(UnitId)

引数	解説
UnitId	ユニットのインデックス。0 からユニット数-1 までの数値です。

解説　　ユニットの名前を返します。

GetUnitParam メソッド

ユニットのパラメータを取得します。

取得　　strParam = AxEng1. GetUnitParam(UnitName)

引数	解説
UnitName	ユニット名。　例）U01

解説　　ユニットのパラメータ文字列を返します。

GetFolderCount メソッド

フォルダ数を取得します。

取得　　lCount = AxEng1. GetFolderCount(UnitName)

引数	解説
UnitName	ユニット名。　例）U01

解説　　現在設定されているフォルダ数を返します。

GetFolderName メソッド

フォルダ名を取得します。

取得　　strName = AxEng1. GetFolderName(UnitName, FolderId)

引数	解説
UnitName	ユニット名。　例）U01
FolderId	フォルダのインデックス。0 からフォルダ数-1 までの数値です。

解説　　フォルダの名前を返します。

GetFolderParam メソッド

フォルダのパラメータを取得します。

取得　　strParam = AxEng1. GetFolderParam(UnitName, FolderName)

引数	解説
UnitName	ユニット名。　例）U01
FolderName	フォルダ名。　例）F01

解説　　フォルダのパラメータ文字列を返します。

GetTagCount メソッド

タグ数を取得します。

取得　　lCount = AxEng1. GetTagCount(UnitName, FolderName)

引数	解説
UnitName	ユニット名。　例）U01
FolderName	フォルダ名。　例）F01

解説　　現在設定されているタグ数を返します。

GetTagName メソッド

タグ名を取得します。

取得　　strName = AxEng1. GetTagName(UnitName, FolderName, TagId)

引数	解説
UnitName	ユニット名。　例）U01
FolderName	フォルダ名。　例）F01
TagId	タグのインデックス。0 からタグ数-1 までの数値です。

解説　　タグの名前を返します。

GetTagParam メソッド

タグのパラメータを取得します。

取得　　strParam = AxEng1. GetTagParam(UnitName, FolderName, TagName)

引数	解説
UnitName	ユニット名。　例）U01
FolderName	フォルダ名。　例）F01
TagName	タグ名。　例）T01

解説　　タグのパラメータ文字列を返します。

GetTagAddress メソッド

タグのアドレスを取得します。

取得　　strAddress = AxEng1. GetTagAddress(UnitName, FolderName, TagName)

引数	解説
UnitName	ユニット名。　例）U01
FolderName	フォルダ名。　例）F01
TagName	タグ名。　例）T01

解説　　タグのアドレス（D0000 や M0000 など）を返します。

ErrCode プロパティ

最新のエラーコードです。

取得　　例）lCode = AxEng1. ErrCode

解説　　最新のエラーコード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 AxEng1.ReadVal("U01.F01.T01", varValue) Then
    txtError.Text = "Fail to read : " & CStr(Eng1.ErrCode)
End If
```

関連項目　　ErrMessage プロパティ

ErrMessage プロパティ

最新のエラーメッセージです。

取得　　例）strMessage = AxEng1. ErrMessage

解説　　最新の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 AxEng1.ReadVal("U01.F01.T01", varValue) Then
    txtError.Text = "Fail to read : " & Eng1.ErrMessage
End If
```

関連項目　　ErrCode プロパティ

エラー関連プロパティ

エラーログを取得します。

ErrLog/ErrLogFileName/ErrLogOverWrite/ErrLogAddDateToFileName/
ErrLogSaveDays/ErrLogSaveDaysTerm

設定

AxEng1. ErrLog = True
AxEng1. ErrLogFileName = “err.txt”
AxEng1. ErrLogOverWrite = False
AxEng1. ErrLogAddDateToFileName = True
AxEng1. ErrLogSaveDays = True
AxEng1. ErrLogSaveDaysTerm = 10　‘10 日以前のファイルが自動削除されます

解説

ErrLog が True の時、ErrLogFileName で指定したファイル名でエラーログを取得します。ErrLogOverWrite プロパティでは、ファイルが既に存在するときに上書きするかどうかを設定します（通常は False です）。

ErrLogAddDateToFileName プロパティでは、エラーログのファイル名の末尾に日付を付加するか否かを指定します。True を指定すると「xxx20030201.txt」のようなファイル名で日付ごとに保存されます。

ErrLogSaveDays プロパティが True の時、ErrLogSaveDaysTerm プロパティ以内の日数より古くなったログファイルを自動削除します。尚、ErrLogSaveDays/ ErrLogSaveDaysTerm プロパティは ErrLogAddDateToFileName プロパティが True の時のみ有効です。これらのプロパティはプロパティページの「エラーログ」から設定することもできます。

関連項目

WriteErrorLog メソッド

WriteErrorLog メソッド

エラーログに任意のメッセージを書き出します。

構文

bResult = AxEng1. WriteErrorLog(Title, Message)

引数	解説
bResult	False - 失敗。True - 成功。
Title	エラーのタイトル。
Message	エラーのメッセージ。

解説

WriteErrorLog メソッドは、手動で独自のメッセージをエラーログファイルに書き出したい場合に利用します。

関連項目

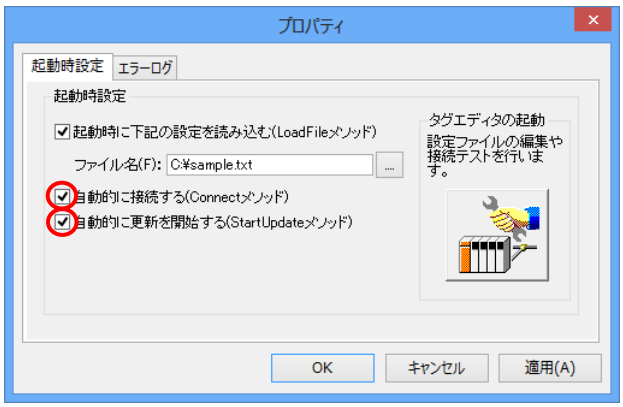
ErrLogFileName/ErrLogOverWrite/ErrLogAddDateToFileName/ErrLogSaveDays/
ErrLogSaveDaysTerm プロパティ

AutoConnect / AutoUpdate プロパティ

起動時に自動的に接続処理/更新処理を行います。

設定　　AxEng1.AutoConnect = True

解説　　ActiveX コントロールがロードされるタイミング（実際はフォームがロードされるタイミング）で、自動的に Connect メソッド/StartUpdate メソッドを呼び出します。これらのプロパティはプロパティページの起動時設定からも設定可能です。



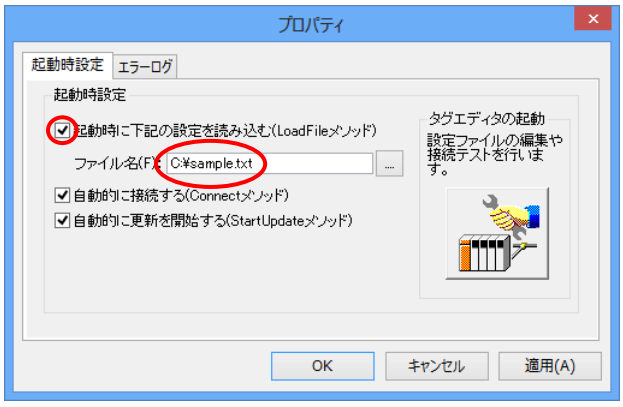
関連項目　　Connect メソッド　StartUpdate メソッド

AutoLoadFile / AutoLoadFileName プロパティ

起動時に自動的に設定ファイルを読み込みます。

設定　　AxEng1.AutoLoadFile = True
AxEng1.AutoLoadFileName = “c:¥sample.txt”

解説　　ActiveX コントロールがロードされるタイミング（実際はフォームがロードされるタイミング）で自動的に LoadFile メソッドを呼び出します。これらのプロパティはプロパティページの起動時設定からも設定可能です。



関連項目　　LoadFile / SaveFile メソッド

StartAccessTimeLog メソッド

アクセスタイムログの出力を開始します。

構文　　bResult = AxEng1. StartAccessTimeLog(FileName, LogNum)

引数	解説
bResult	False - 失敗。True - 成功。
FileName	ログを保存するファイル名。
LogNum	ログを取る数。

解説　　アクセスタイムログの出力を開始する。ログが LonNum になると自動的に終了します。StopAccessTimeLog で強制的にログ出力を終了することができます。

関連項目　　StopAccessTimeLog イベント

StopAccessTimeLog メソッド

アクセスタイムログを終了する。

構文　　bResult = AxEng1. StopAccessTimeLog()

引数	解説
bResult	False - 失敗。True - 成功。

解説　　アクセスタイムログを終了します。ログを開始後はログ数が LogNum になると自動終了されますので、通常はこのメソッドは使いません。処理途中で強制的に終了させたい場合などに使用します。

関連項目　　StartAccessTimeLog イベント

ExecuteTagEditer メソッド

タグエディタを起動する。

構文　　bResult = AxEng1. ExecuteTagEditer (FileName)

引数	解説
bResult	False - 失敗。True - 成功。
FileName	設定ファイル名。

解説　　タグエディタを起動します。

5－3　イベント

ValueStateChanged イベント

値及び値の状態が変化したときに発生するイベントです。

構文

●VB.Net

```
Private Sub AxEng1_ValueStateChanged(sender As Object,
  e As AxFAENGINELib5._DFAEngineEvents_ValueStateChangedEvent)
    Handles AxEng1.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 Eng1_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します。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は、正常に通信ができた最新値が Value に格納されています。

関連項目

ValueChanged イベント

ValueChanged イベント

値が変化したときに発生するイベントです。

構文

●VB.Net

```
Private Sub AxEng1_VisibleChanged(sender As Object,
  e As System.EventArgs) Handles AxEng1.VisibleChanged
```

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 Eng1_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します。Value 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。またデッドバンドを設定することも可能です。設定する場合は下記のダイアログからデッドバンド設定を行います。



左の例では、値の変化が 10 以下の場合は、イベントが発生しません。

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イベントです。

構文

●VB.Net

```
Private Sub AxEng1_ValueBitOn(sender As Object,
  e As AxFAENGINELib5._DFAEngineEvents_ValueBitOnEvent)
    Handles AxEng1.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 Eng1_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 した時に発生します。Value 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 が Off でそれ以外を On と解釈します。文字列の場合は、一度、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

ValueOverLimit イベント

上下限値を超えたときに発生するイベントです。

構文

●VB.Net

```
Private Sub AxEng1_ValueOverLimit(sender As Object,
  e As AxFAENGINELib5._DFAEngineEvents_ValueOverLimitEvent)
    Handles AxEng1.ValueOverLimit
```

引数	解説
e.tagPath	タグパスです。 例）U01.F01.T01
e.value	値です。
e.area	現在のタグの値がどこの領域にあるかをあらわす 0～5 までの数。

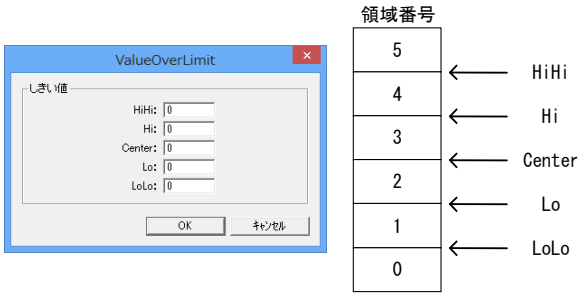
●VB6.0

```
Private Sub Eng1_ ValueOverLimit
  (ByVal TagPath As String, ByVal Value As Variant, ByVal Area As Long)
```

引数	解説
TagPath	タグパスです。 例）U01.F01.T01
Value	値です。
Area	現在のタグの値がどこの領域にあるかをあらわす 0～5 までの数。

解説

上下限値を超えたときに発生します。このイベントを発生させるには、フィルタの「イベント指定」を追加し、下記のダイアログから上下限の設定を行う必要があります。



Area は、「しきい値設定」によって指定された値によって、左のように割り当てられます。

ValueRapidChanged イベント

値に急速な変化があったときに発生するイベントです。

構文

●VB.Net

```
Private Sub AxEng1_ValueRapidChanged(sender As Object,
  e As AxFAENGINELib5._DFAEngineEvents_ValueRapidChangedEvent)
    Handles AxEng1.ValueRapidChanged
```

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 Eng1_ ValueRapidChanged
  (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

タグの値が急激な変化をしたときに発生します。このイベントを発生させるには、フィルタの「イベント指定」を追加し、下記のダイアログから設定を行う必要があります。



左の例では、6000 ミリ秒＝6 秒以内に、タグの値が 10 以上変化したときに、イベントが発生します。6 秒以内に 10 未満しか変化しなかった場合や、6 秒以上かかって 10 以上変化した場合にはイベントは発生しません。

ValueInitialize イベント

初回のキャッシュ初期化時のみ発生するイベントです。

構文

●VB.Net

```
Private Sub AxEng1_ValueInitialize(sender As Object,
  e As AxFAENGINELib5._DFAEngineEvents_ValueInitializeEvent)
    Handles AxEng1.ValueInitialize
```

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 Eng1_ ValueInitialize
  (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

通信開始後、はじめてキャッシュが初期化されるときのみ発生します。

ValueRefresh イベント

自動更新のためのリフレッシュ通信が完了するごとに発生するイベントです。

構文

●VB.Net

```
Private Sub AxEng1_ValueRefresh(sender As Object,
  e As AxFAENGINELib5._DFAEngineEvents_ValueRefreshEvent)
    Handles AxEng1.ValueRefresh
```

引数	解説
e.folderPath	フォルダパス。 例）U01.F01
e.result	True – 成功 False – 失敗。

●VB6.0

```
Private Sub Eng1_ ValueRefresh
  (ByVal FolderPath As String, ByVal Result As Boolean)
```

引数	解説
FolderPath	フォルダパス。 例）U01.F01
Result	True – 成功 False – 失敗。

解説

自動更新のための通信が完了するごとに発生します。このイベント内では、更新後の値を GetVal メソッドにより高速に取得することができます。自動更新は一定周期間隔で行われるため、定期的なデータログを取りたい場面などではこのイベントが利用できます。また、ReadRefresh メソッドを呼び出した後にも発生します。

StateChanged イベント

ユニットのデバイスダウンの状態が変化した時に発生するイベントです。

構文

●VB.Net

```
Private Sub AxEng1_StateChanged(sender As Object,
  e As AxFAENGINELib5._DFAEngineEvents_StateChangedEvent)
    Handles AxEng1.StateChanged
```

引数	解説
e.unitName	ユニット名。　例）U01
e.state	False – 接続状態 NG　True – 接続状態 OK

●VB6.0

```
Private Sub Eng1_StateChanged
  (ByVal UnitName As String, ByVal State As Boolean)
```

引数	解説
UnitName	ユニット名。　例）U01
State	False – 接続状態 NG　True – 接続状態 OK

解説

初回のイベント発生として、一回目の自動更新の結果が送られてきます。以降は、ユニットのデバイスダウン状態が変化した都度イベントが発生します。
尚、初回から失敗している場合の動作としては、まず、初回の自動更新直後に False でイベントが発生し、その後、正式にデバイスダウンとみなされた時に再度 False でイベントが発生します。

関連項目

GetUnitState メソッド

ErrorOccured イベント

エラーが発生したときに発生するイベントです。

構文

●VB.Net

```
Private Sub AxEng1_ErrorOccured(sender As Object,
  e As AxFAENGINELib5._DFAEngineEvents_ErrorOccuredEvent)
    Handles AxEng1.ErrorOccured
```

引数	解説
e.code	エラーコード
e.message	エラーメッセージ

●VB6.0

```
Private Sub Eng1_ ErrorOccured
  (ByVal Code As Long, ByVal Message As String)
```

引数	解説
Code	エラーコード
Message	エラーメッセージ

解説　自動更新や各メソッド呼び出し等、全ての通信においてエラーが発生したときに呼び出されるイベントです。

関連項目　ErrCode プロパティ ErrMessage プロパティ

EndAccessTimeLog イベント

アクセスタイムログが終了したときに発生するイベントです。

構文

●VB.Net

```
Private Sub AxEng1_EndAccessTimeLog(sender As Object,
  e As AxFAENGINELib5._DFAEngineEvents_EndAccessTimeLogEvent)
    Handles AxEng1.EndAccessTimeLog
```

引数	解説
e.fileName	アクセスタイムログのファイル名

●VB6.0

```
Private Sub Eng1_ EndAccessTimeLog(ByVal FileName As String)
```

引数	解説
FileName	アクセスタイムログのファイル名

解説　アクセスタイムログが終了したときに呼び出されます。

関連項目　StartAccessTimeLog / StopAccessTimeLog メソッド

5－4　Ver3 との比較

下表は、Ver3 と Ver5 のインターフェースの比較です。

	Ver3	Ver4
■通信		
	Connect	Connect
	DisConnect	DisConnect
	ConnectAll	Connect
	DisConnectAll	DisConnect
	Start	StartUpdate
	Stop	StopUpdate
	StartAll	StartUpdate
	StopAll	StopUpdate
	ReadVal	ReadVal
	WriteVal	WriteVal
	Value	GetVal/SetVal
	ReadRefresh	ReadRefresh
	WriteRefresh	WriteRefresh
	ConnectTest	AliveTest
	SetEngine	AccessPath(原則として廃止)
	NowEngineName	AccessPath(原則として廃止)
■タグ登録・解除		
	AddEngine	AddUnit
	DelEngine	DelUnit
	AddFolder	AddFolder
	DelFolder	DelFolder
	AddTag	AddTag
	DelTag	DelTag
	ClearAll	ClearAll
	LoadFile/SaveFile	LoadFile/SaveFile
■タグ参照		
	EngNum	GetUnitCount
	GetFirstEngName	GetUnitName
	GetNextEngName	GetUnitName
	GetEngParmStr	GetUnitParam
	FolderNum	GetFolderCount
	GetFirstFolderName	GetFolderName
	GetNextFolderName	GetFolderName
	GetFolderParmStr	GetFolderParam
	TagNum	GetTagCount
	GetFirstTagName	GetTagName
	GetNextTagName	GetTagName

＜次頁につづく＞

<前頁のつづき>

	Ver3	Ver4
■タグ参照		
	GetTagParmStr	GetTagParam
	GetTagDeviceStr	GetTagAddress
■エラー処理		
	ErrCode	ErrCode
	ErrCodeSub	ErrMessage
	ErrLog	ErrLog
	ErrLogFileName	ErrLogFileName
	GetErrLogNum	廃止
	ClearErrLog	エラーログ自動削除機能
	WriteErrorLog	WriteErrorLog
■起動時設定		
	AutoConnect	AutoConnect
	AutoStart	AutoUpdate
	AutoReadFile	AutoReadFile
	AutoReadFileName	AutoReadFileName
■その他		
	StartAccessTimeLog	StartAccessTimeLog
	StopAccessTimeLog	StopAccessTimeLog
	ExecuteTagEditer	ExecuteTagEditer
	EngRun	U01!Update
	Debug	U01!Virtual
	AsciiOrder	ユニットプロパティ
	BCDFlagType	ユニットプロパティ
	SetAuto	F01!Update
	SetManual	F01!Update
	StartAddr	廃止
	MoveStartAddr	F01!OffsetTag
	MoveStartAddrBit	F01!OffsetBit
	ResetMaxMin	T01!ResetFilter
■イベント		
	ValueChanged	ValueChanged
	BitOn	ValueBitOn
	BitOff	ValueBitOff
	OverLimit	ValueOverLimit
	RapidChange	ValueRapidChanged
	StartEvent	ValueInitialize
	DataIn	ValueRefresh 内で、GetVal にて取得
	AccessDone	ValueRefresh
	StartEngine	廃止
	ConnectEngine	廃止
	ErrorOccured	ErrorOccured
	EndAccessTimeLog	EndAccessTimeLog

５－５　Ver4 との比較

FA-Engine5 は FA-Engine4 とまったく同等のインターフェースになっています。
バージョンを Ver4 から Ver5 に変更する場合、FA-Engine(ActiveX）を貼り付けなおすのみで、プログラムなどは変更する必要がありません。

５－６　エラーコード

エラーコードは大きく２つに分けられます。

1000 番：　通信以外のエラーです。プログラムや設定に問題がある可能性があります。
1001 番以降：　通信エラーです。プログラムではなく、アドレス設定や通信環境に問題がある可能性があります。

エラーの詳細は ErrMessage プロパティ、もしくは ErrorOccured イベントの Message 引数で確認して下さい。

コード	説明
1000	通信以外のエラーです。主にプログラムや設定ミスにより発生します。
1001	接続で問題が発生しました。PLC となんらかの理由で接続できなかった時に発生します。接続パラメータや PLC の状態を再度確認して下さい。
1002	タグのパラメータに誤りがあります。タグの設定を確認して下さい。
1003	通信を試みたが結果を得られなかった場合に発生します。
1004	通信を試みたがタイムアウトになりました。通信環境をチェックして下さい。
1005	壊れたパケットを受け取りました。Ethernet 接続の場合、タイムアウト時間を延ばして下さい。
1006	PLC からエラーコードが返ってきました。詳しい内容はエラーメッセージを参照して下さい。
1007	通信先からビジー応答が返ってきました。
1008	通信リクエストキャンセルにより通信要求を打ち切ります。
1009	メモリーが不足しています。

「FA-Engine ユーザーズマニュアル」Copyright © 2010 RoboticsWare, Inc. All rights reserved.
2010 年 6 月 1 日 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。