IVI 計測器ドライバ
プログラミングガイド
Excel VBA 編

April 2012 Revision 2.2

目次

はじめに

本ガイドでは、KikusuiPwr IVI 計測器ドライバ（KIKUSUI PWR-01 シリーズ直流電源）を使用する例を示します。他社メーカーおよび他機種用の IVI 計測器ドライバでも、ほぼ同様の手順で使用できます。

本ガイドでは、Microsoft Excel for Microsoft 365 MSO (16.0.13801.20288)64 ビットバージョン 2102 を使用し、Windows10（x64）上で動作する 64bit（x64）プログラムを作成する場合を例に説明します。

Excel VBA で使用する計測器ドライバ

Excel VBA は、IVI-COM 計測器ドライバを使うのに適した開発環境の一つです。ActiveX コントロールのような COM オブジェクトを使うプログラミング手法が Excel VBA では一般的ですが、IVI-COM 計測器ドライバでも COM オブジェクトを使う方法と同様な手順でプログラムを作成できます。

本ガイドでは、IVI-COM 計測器ドライバを使用してプログラミングする手順を例に説明しています。

使用できるインターフェース

IVI 計測器ドライバは以下の 2 つのインターフェースに対応しています。

- **スペシフィック・インターフェース**
  計測器ドライバに固有のインターフェースです。使用する計測器の機能を最大限に利用できます。
- **クラス・インターフェース**
  IVI 仕様書で定義されている計測器クラスのインターフェースです。
  インターチェンジャビリティ機能を利用できますが、機種固有の機能を使うことは制限されます。

本ガイドでは、それぞれのインターフェースを使用してプログラミングする方法について説明します。

Memo

- 計測器ドライバが所属する計測器クラスは、ドライバの Readme.txt に記載されています。Readme 文書は、［Start］ボタン＞［Kikusui］＞［KikusuiPwr IVI Driver 1.0.0 Documentation］メニューから確認できます。
- 計測器ドライバがどの計測器クラスにも属していない場合、クラス・インターフェースを利用できないため、インターチェンジャビリティ機能を利用するアプリケーションは作成できません。

事前にインストールが必要なソフトウェア

プログラミングの前に、下記のソフトウェアを当社ウェブサイトのダウンロードサービス（https://www.kikusui.co.jp/download/）からダウンロードしてインストールしてください。

- KI-VISA Ver5.5 以降
- PWR-01 seriesIVI-COM 多重環境計測器ドライバ Ver1.0 以降

詳細は、「IVI 計測器ドライバ プログラミングガイド セットアップ編」を参照してください。

クラス・インターフェースを使用する場合には、下記のソフトウェアもインストールが必要です。

- NI-MAX Ver20.5.0 以降
- IVI compliance Package Ver20.0.0 以降

VBA を使用する準備

Excel シートに直接ボタンを配置するアプリケーションを例に説明します。Excel でマクロを有効にして、新規のマクロを作成します。

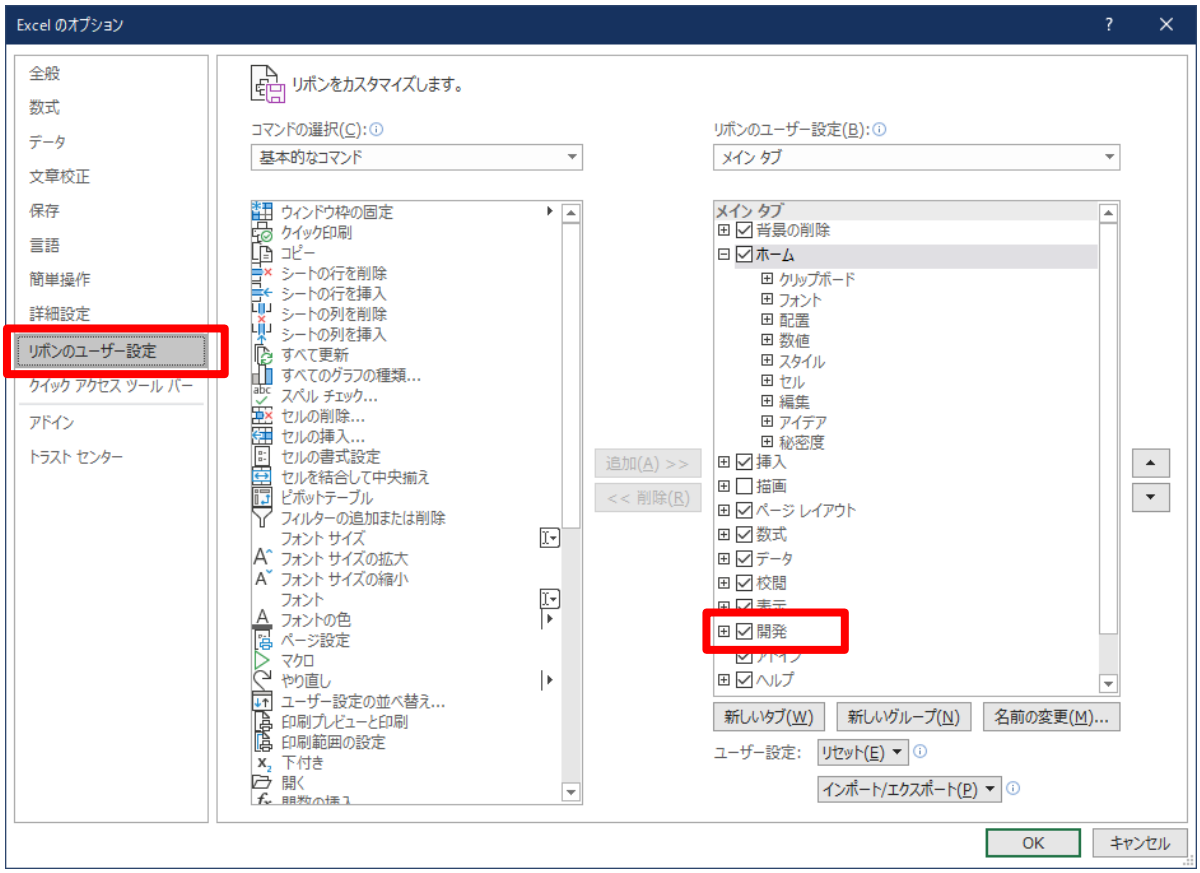
Excel でマクロを有効にする

VBA を使用するには、Excel でマクロを有効にします。

1 **Excel を起動し、［ファイル］タブをクリックします。**

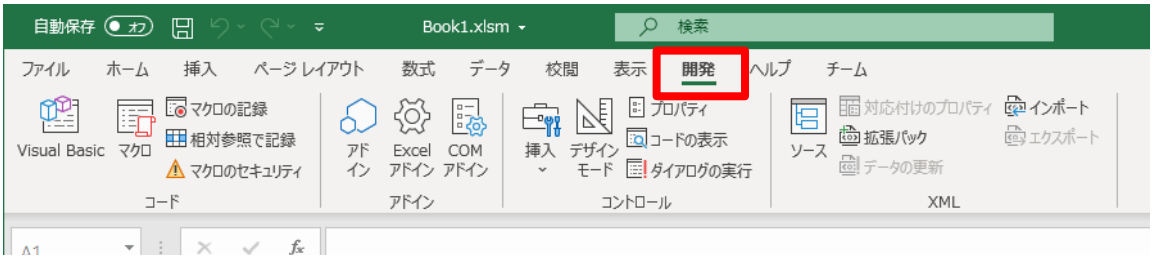
2 **画面左下の［オプション］をクリックします。**

3 **［リボンのユーザー設定］＞［開発］をチェックします。**



リボンメニューに［開発］タブが表示されます。

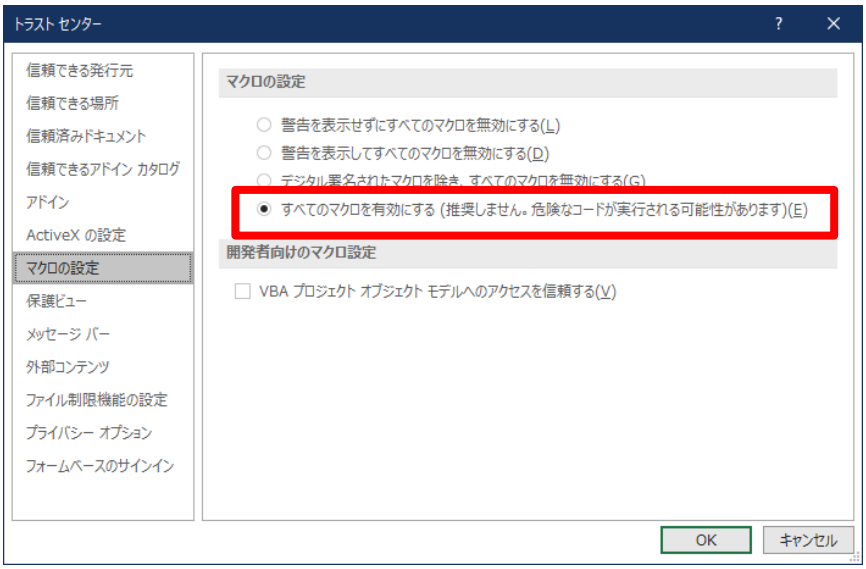
4 リボンメニューで［開発］タブをクリックします。



5 ［コード］の［マクロのセキュリティ］をクリックします。

トラストセンターダイアログが表示されます。

6 ［マクロの設定］＞［すべてのマクロを有効にする］をチェックします。



7 ［OK］をクリックします。

新規のマクロを作成する

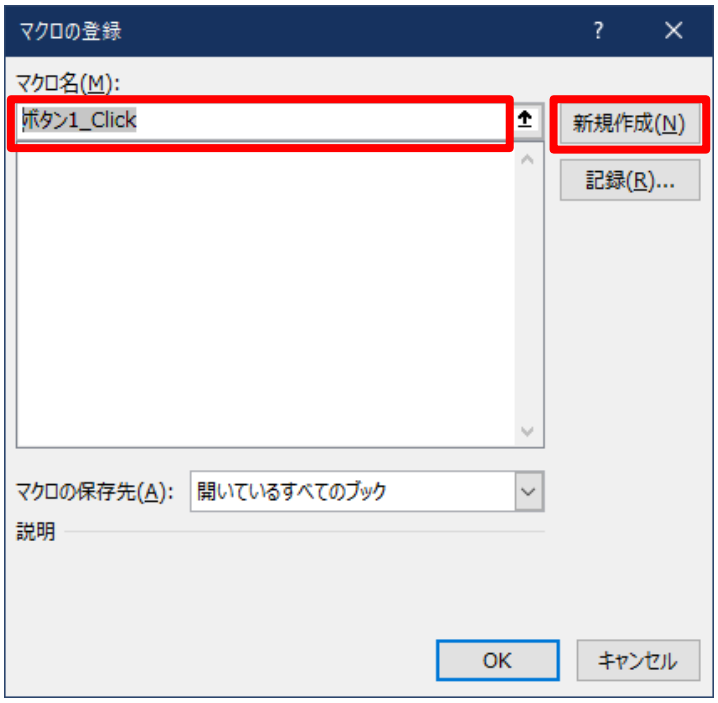
Excel でボタンを作成して新規のマクロを登録します。

1 **［開発］タブの［挿入］＞［フォームコントロール］＞［ボタン（フォームコントロール）］をクリックします。**

2 **ボタンを置きたいセルの位置をクリックしてボタンを作成します。**

マクロの登録ダイアログが表示されます。

3 **任意のマクロ名を入力して［新規作成］をクリックします。**



Visual Basic のエディタが表示されます。エディタには、ボタンハンドラのソースコードが表示されます。以降は、ボタンハンドラ内にプログラムを記述します。

スペシフィック・インターフェースを使用したプログラミング

スペシフィック・インターフェースを使用すると、計測器ドライバの機種固有機能を最大限に利用できます。

ここでは、スペシフィック・インターフェースを使用してプログラムする手順を説明します。

Memo

スペシフィック・インターフェースの場合、インターチェンジャビリティ機能は利用できません。インターチェンジャビリティ機能を利用する場合には、クラス・インターフェース（p.7）を使用してください。

プログラミングの準備

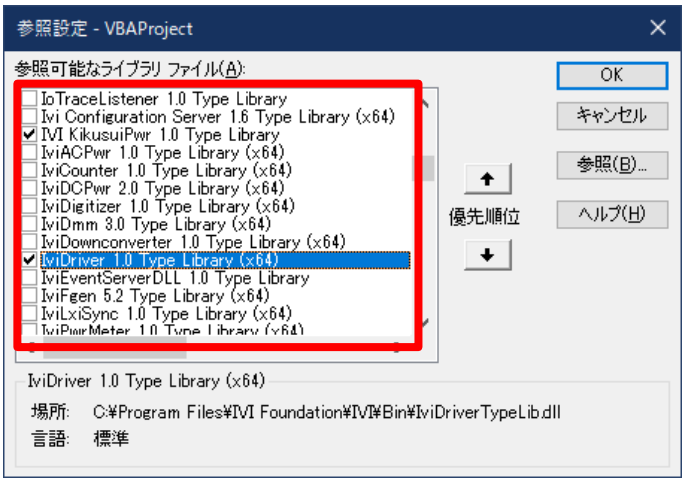
タイプライブラリをインポートする

使用する IVI-COM 計測器ドライバのタイプライブラリをインポートします。

1 **Visual Basic で［ツール］＞［参照設定］をクリックします。**

参照設定ダイアログが表示されます。

2 **［IVI KikusuiPwr 1.0 Type Library］と［IviDriver 1.0 Type Library］をチェックします。**



3 **［OK］をクリックします。**

プログラムを構成する

オブジェクトの作成とセッションの開始／終了

「新規のマクロを作成する」（p.7）で開いたボタンハンドラのソースコードに、計測器ドライバオブジェクトの作成とセッションをオープン、クローズする下記のサンプルコードを記述します。ここでは、IP アドレス 192.168.1.5 を持つ計測器（Kikusui PWR-01 シリーズ直流電源）が LAN で接続されていると仮定します。

```
Sub CommandButton1_Click()
      Dim inst As IKikusuiPwr01
      Set inst = New KikusuiPwr01

      inst.Initialize "TCPIP::192.168.1.5::INSTR", True, True, ""

      inst.Close
End Sub
```

上記の記述だけでも、プログラムを実行することが可能です。動作確認のためにプログラムを実行する場合には、「プログラムを実行する」（p.14）を参照してください。

Initialize メソッドのパラメータ

すべての IVI-COM 計測器ドライバには、IVI 仕様書で定義された Initialize メソッドがあります。Initialize メソッドには、下記のパラメータがあります。

パラメータ	タイプ	説明
ResourceName	String	計測器が接続されている I/O インターフェース、アドレスなどによって決定される VISA リソース名の文字列です。たとえば、IP アドレス 192.168.1.5 の計測器を LAN で接続し、VXI-11 インターフェースで制御している場合には、「TCPIP::192.168.1.5::INSTR」となります。
IdQuery	Boolean	TRUE を指定した場合、計測器に対して "*IDN?" クエリなどを発行して機種情報を問い合わせます。
Reset	Boolean	TRUE を指定した場合、"*RST" コマンドなどを発行して計測器の設定をリセットします。
OptionString	String	IVI 定義の設定を変更できます。

OptionString を設定するには

Initialize メソッドの OptionString では、下記の IVI 定義を設定できます。

IVI 定義	デフォルト値
RangeCheck	TRUE
Cache	TRUE
Simulate	FALSE
QueryInstrStatus	FALSE
RecordCoercions	FALSE
Interchange Check	FALSE

OptionString は文字列のパラメータです。下記に、サンプルを記載します。

```
QueryInstrStatus = TRUE , Cache = TRUE , DriverSetup=12345
```

書式について、次の点に注意してください。

- 設定値を指定しない場合にはデフォルト値が適用されます。
- 設定値は Boolean 型です。「TRUE」、「FALSE」、「1」、「0」のいずれかが設定できます。
- 機能名称および設定値はケース・インセンシティブです。大文字と小文字は区別されません。
- 複数の項目を設定する場合には、カンマで区切ります。
- 計測器ドライバによっては、「DriverSetup」パラメータがサポートされています。
  「DriverSetup」は、IVI 仕様書では定義されない項目を Initialize の呼び出し時に指定するパラメータです。利用目的や書式はドライバに依存します。そのため、「DriverSetup」を設定する場合には、「Option String」の最後の項目として指定します。「DriverSetup」の指定内容については、ドライバの Readme またはオンライン・ヘルプなどを参照してください。

チャンネルと試験条件を設定する

電源装置などの場合、IVI 計測器ドライバは複数のチャンネルが装備されていることを前提に設計されています。そのため、計測器のパネル設定について操作をするプロパティやメソッドは、IVI 仕様書でリピーテッド・キャパビリティ（一般的な COM の用語ではコレクション）と呼ばれるオブジェクト配列の概念を実装するケースが多く見られます。直流電源の計測器ドライバの場合には、Output コレクションが該当します。

KikusuiPwr IVI-COM ドライバの場合には、IKikusuiPwr01Outputs と IKikusuiPwr01Output があります。複数形のオブジェクトがコレクションで、その中に単数形のオブジェクトが 1 個以上含まれます。一般的に、両者は複数形、単数形の違いを除いて同じ名称が使われます。

下記は、「プログラムを構成する」（p.9）のコードに Kikusui PWR-01 シリーズ直流電源の「OutputChannel1」という名前の出力チャンネル制御を加えたサンプルコードです。

```
Sub CommandButton1_Click()
      Dim inst As IKikusuiPwr01
      Set inst = New KikusuiPwr01

      inst.Initialize "TCPIP::192.168.1.5::INSTR", True, True, ""

      Dim output As IKikusuiPwr01Output
      Set output = inst.Outputs.Item("OutputChannel1")
      output.VoltageLevel = 20.0
      output.CurrentLimit = 2.0
      output.Enabled = True
      inst.Close
End Sub
```

サンプルコードの解説

IKikusuiPwr01 インターフェースの Outputs プロパティを通じて IKikusuiPwr01Outputs を取得しています。更に、Item プロパティを使って IKikuauiPwr01Output インターフェースを取得しています。

VoltageLevel プロパティは、電圧レベルを設定します。CurrentLimit プロパティは、電流リミットを設定します。Enabled プロパティは、出力の ON ／ OFF を設定します。

Item プロパティに渡すパラメータ（サンプルでは "OutputChannel1"）には、参照したい単品の Output オブジェクト名を指定します。このオブジェクト名はドライバごとに異なります。特定の計測器ドライバで使用できる名前は、通常はドライバのヘルプなどに記載されています。後述するコードを書くことでも調べることができます。

計測器ドライバ固有のオブジェクト名を調べる

計測器固有のオブジェクト名がわからない場合には、Output コレクションの Count、Name（いずれもリードオンリー）を利用して調べることができます。

下記に、サンプルコードを記載します。

```
Dim outputs As IKikusuiPwr01Outputs
Set outputs = inst.outputs

Dim n As Integer
Dim c As Integer
c = outputs.Count

For n = 1 To c
      Dim name As String
      name = outputs.name(n)
      Debug.Print name
Next
```

Count プロパティは、コレクションが持つ単品オブジェクトの個数を返します。Name プロパティは、与えられたインデックス番号に該当する単品オブジェクトの名前を返します。

Name プロパティで返ってきた名前が、「チャンネルと試験条件を設定する」（p.11）で Item プロパティに渡すパラメータになります。

上記のサンプルコードでは、For/Next を使ってインデックス 1 から Count までを反復処理しています。Name パラメータに渡すインデックス番号は 0 ベースではなく 1 ベースであることに注意してください。

エラーを処理する

範囲外の値をプロパティに渡したり、サポートされていない機能を呼び出したりすると、計測器ドライバにエラーが発生します。IVI-COM 計測器ドライバでは、計測器ドライバ内で発生したエラーは COM 例外としてクライアントプログラムに伝えられます。VBA の場合には、COM 例外は On Error Goto ステートメントを使って処理できます。

下記は、「チャンネルと試験条件を設定する」（p.11）のコードにエラー処理を加えたサンプルコードです。

```
Sub CommandButton1_Click()
      On Error GoTo DRIVER_ERR:
      Dim inst As IKikusuiPwr01
      Set inst = New KikusuiPwr01Lib.KikusuiPwr01

      inst.Initialize "TCPIP::192.168.1.5::INSTR", True, True, ""

      Dim output As IKikusuiPwr01Output
      Set output = inst.Outputs.Item("OutputChannel1")
      output.VoltageLevel = 20.0
      output.CurrentLimit = 2.0
      output.Enabled = True

      inst.Close

      Exit Sub
DRIVER_ERR:
      Debug.Print Err.Description
End Sub
```

サンプルコードの解説

このコードでは、On Error Goto ステートメントを使ってエラーを処理しています。たとえば、Item プロパティに渡した名前が間違っている場合、VoltageLevel に設定する値が適正範囲から外れている場合、または計測器との通信に失敗した場合には、いずれも計測器ドライバ内で COM 例外が発生します。上記のコードでは、例外が発生した場合に簡単なメッセージをイミディエイトウインドウに表示します。

プログラムを実行する

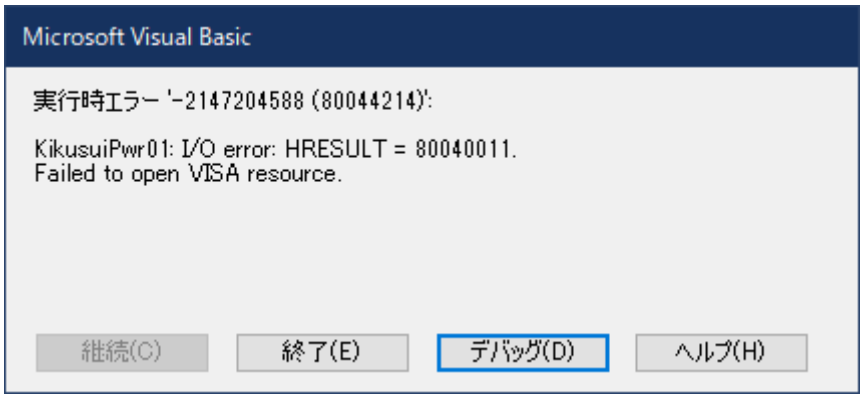
構成したプログラムを実行します。

1 **Visual Basic のワークシート上に作成したボタンをクリックします。**

VBA マクロが動作し、プログラムが終了します。

実行に失敗した場合

COM 例外（VBA ランタイム・エラー）が発生します（p.13）。



クラス・インターフェースを使用したプログラミング

クラス・インターフェースを使用してプログラムする手順を説明します。クラス・インターフェースでは、IVI 仕様で定義された計測器クラスのインターフェースを使用してプログラミングすることで、計測器クラス・インターフェースを利用したインターチェンジャビリティを実現できます。インターチェンジャビリティを利用すると、アプリケーションを再度コンパイル・リンクすることなく計測器を交換できます。

Memo

- インターチェンジャビリティを利用するには、交換前後の両機種に対して IVI-COM 計測器ドライバが提供されており、かつそれらのドライバが同じ計測器クラスに属している必要があります。異なる計測器クラス間でのインターチェンジャビリティは実現できません。
- クラス・インターフェースを使用したプログラミングでは、使用できる機種固有の機能が制限されます。機種固有の機能を最大限に使用するには、スペシフィック・インターフェース（p.8）を使用してプログラミングしてください。

プログラミングの準備

仮想インストルメントを作成する

インターチェンジャビリティ機能を利用するアプリケーションを作成するには、あらかじめ仮想インストルメントを作成する必要があります。

Memo

インターチェンジャビリティ機能を損なうため、アプリケーション・コード内に特定の IVI-COM 計測器ドライバに依存した記述（例：KikusuiPwr 型で直接オブジェクトを生成）や、特定 VISA アドレス（リソース名）の記述（例："TCPIP::192.168.1.5::INSTR"）などを行わないでください。

IVI 仕様では、計測器ドライバとアプリケーションの外部に IVI コンフィグレーション・ストアを置くことでインターチェンジャビリティ機能を実現します。

アプリケーションは機種固有の計測器ドライバを直接使うのではなく、計測器クラス・ドライバと呼ばれる特別な計測器ドライバを通じて制御します。

制御の際は、IVI コンフィグレーション・ストアの内容に従って計測器ドライバの DLL を選択し、間接的にロードされた計測器ドライバに、機種に依存しないクラス・ドライバの関数を通じてアクセスします。

IVI コンフィグレーション・ストアには、XML ファイル（C:/ProgramData/IVI Foundation/IVI/IviConfigurationStore.xml）が使用されます。主に IVI 計測器ドライバや一部の VISA/IVI コンフィグレーション・ツールが、IVI Configuration Server DLL を通じてアクセスします。アプリケーションが利用することは通常はありません。

VBA を使用する場合には、National Instruments 社製のソフトウェア NI-MAX（NIMeasurement and Automation Explorer）を使用して IVI ドライバのコンフィグレーション行います。

以降、NI-MAX を使用して仮想インストルメントを作成する手順について説明します。

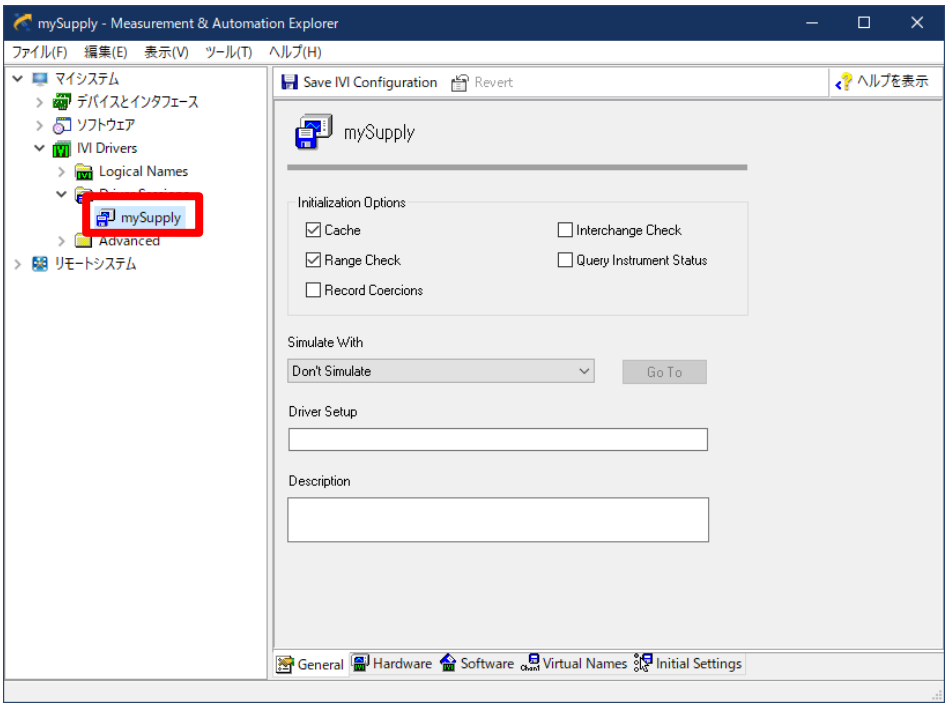
Driver Sessions を作成する

1 **NI-MAX を起動し、画面左のツリー表示で［IVI Drivers］以下の階層を確認します。**

2 **［Driver Sessions］を右クリックし、［Create New（case-sensitive）］を選択します。**

3 **Driver Sessions の名前を設定します。**

ここでは［mySupply］とします。

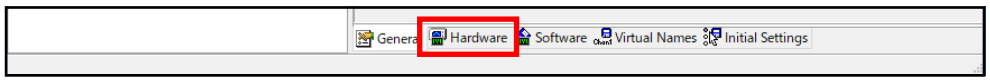


Driver Sessions が作成されました。

Hardware Asset を作成する

Hardware Asset（ハードウェア・アセット）では、使用する計測器をどのような経路で接続するかを指定します。

1 **［Hardware］タブを選択します。**



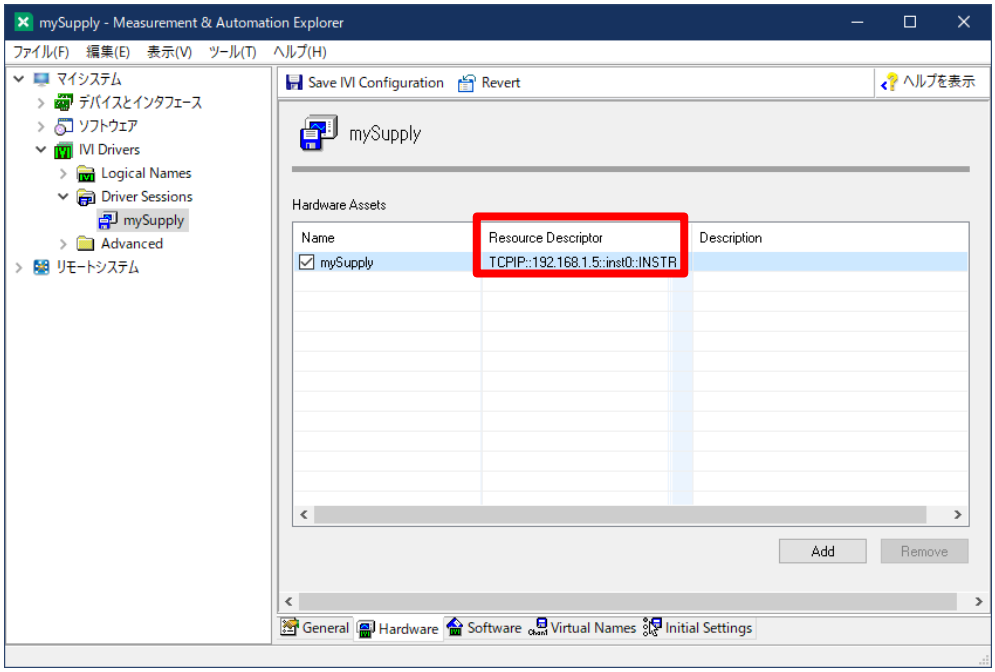
2 **［Add］をクリックして Hardware Asset を新規作成します。**

3 **Hardware Asset の名前を設定します。**

ここでは［mySupply］とします。

4 **［Resource Descriptor］に、使用する計測器が接続されている VISA アドレスを指定します。**

この例では、「TCPIP::192.168.1.5::inst0::INSTR」を指定しています。

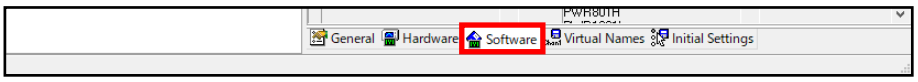


Hardware Asset が作成されました。

Software Module を設定する

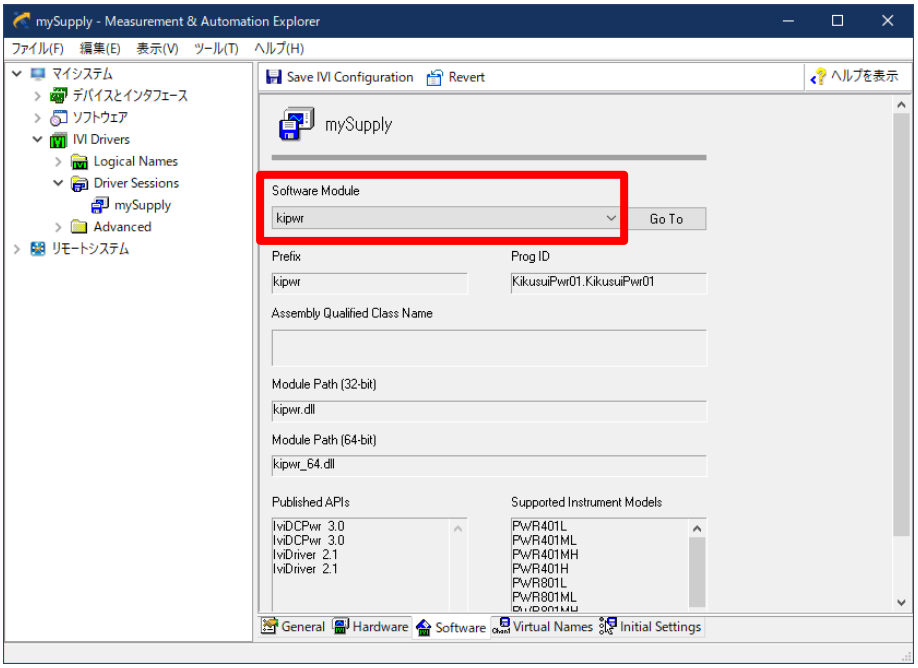
Software Module（ソフトウェア・モジュール）では、計測器ドライバモジュール（DLL モジュール）を設定します。

1 **［Software］タブを選択します。**



2 **［Software Module］のリストから、使用する計測器ドライバモジュールを選択します。**

この例では、［kipwr］を選択しています。

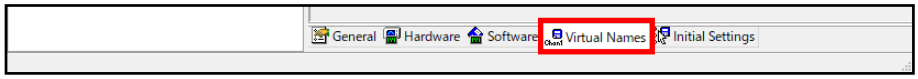


Software Module が設定されました。

Virtual Name を作成する

Virtual Name（バーチャルネーム）では、計測器ドライバのチャンネル名を仮想化した名称を作成します。チャンネルの指定が必要な計測器ドライバの場合、有効なチャンネル名は計測器ドライバによって異なるためです。

1 **［Virtual Names］タブを選択します。**



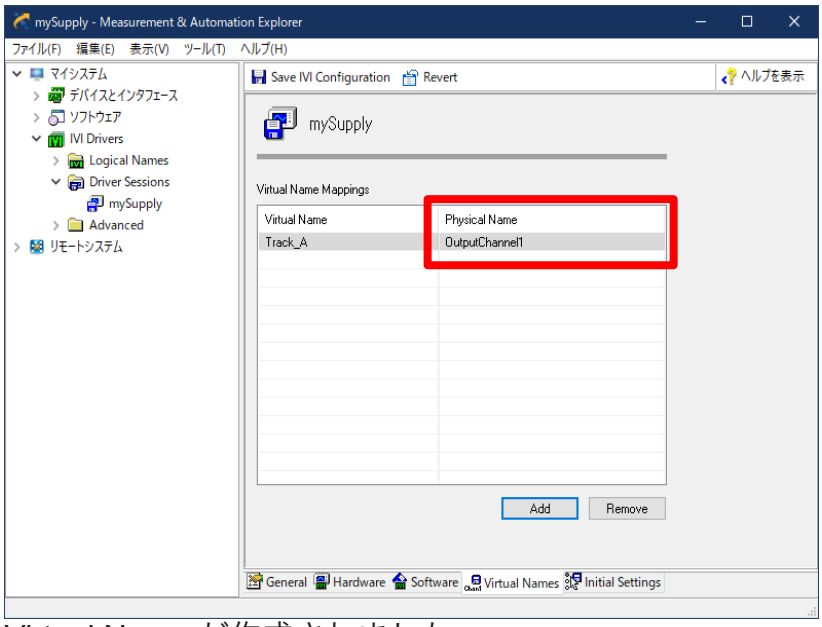
2 **［Add］をクリックしてバーチャルネームを追加し、［Virtual Name］に［Track_A］を入力します。**

［Physical Name］リストには、計測器が稼動するチャンネルが表示されます。

3 **［Physical Name］の列で、リストに表示されているチャンネル名を選択するか、または有効なチャンネル名を入力します。**

この例では、［OutputChannel1］を選択または入力します。

> **Memo**
>
> ドライバの実装状態や、多チャンネル電源装置での構成によっては、すべてのチャンネル名が表示されない場合があります。ドライバの利用可能なチャンネル名については、ドライバごとの Readme 文書またはオンライン・ヘルプなどを参照してください。



Virtual Name が作成されました。

Logical Name を設定する

Logical Name（ロジカル・ネーム）とは、NI-MAX で設定される仮想計測器の名称です。

1 画面左のツリー表示で［**IVI Drivers**］以下の階層を確認します。

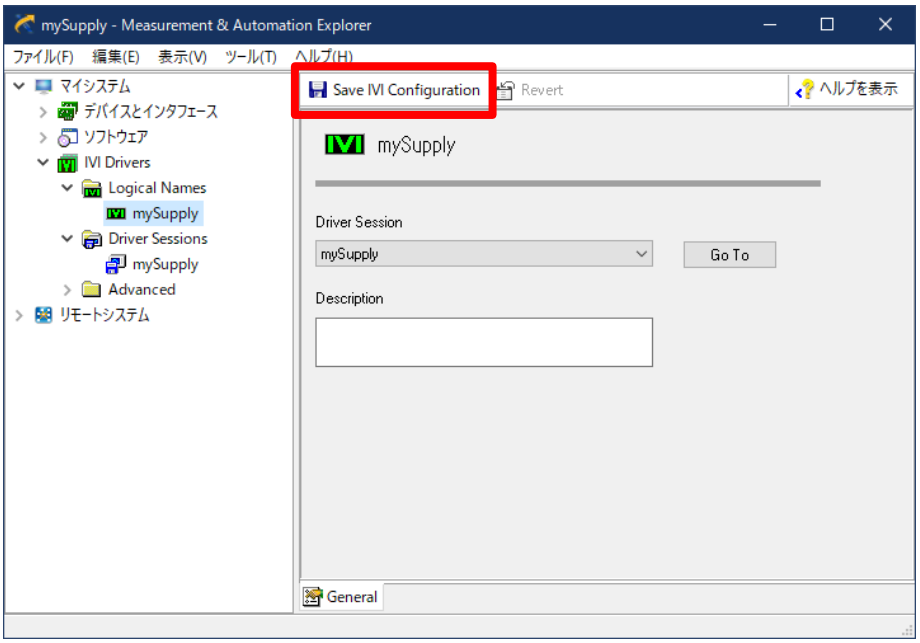
2 ［**Logical Name**］を右クリックして［**Create New（case-sensitive）**］を選択します。

3 **Logical Name** の名前を設定します。

ここでは［mySupply］とします。

4 ［**Driver Session**］で［**mySupply**］を選択します。

5 ツールバーの［**Save IVI Configuration**］をクリックし、設定を保存します。



Logical Name が設定されました。

以上で、仮想インストルメントの作成は完了です。

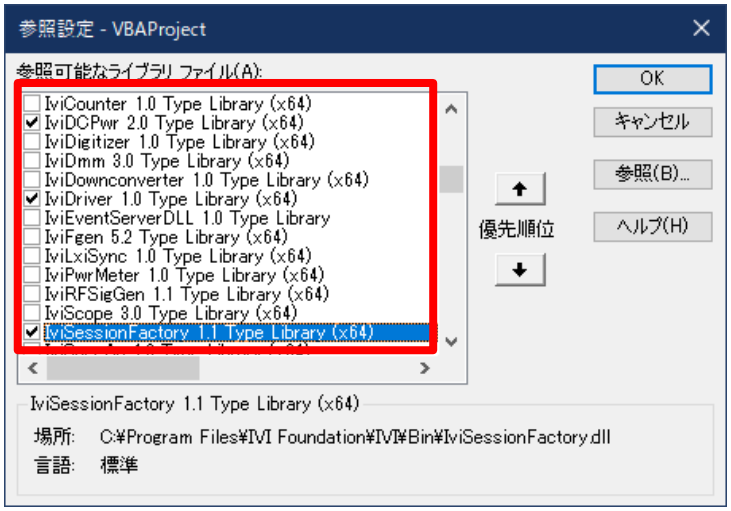
タイプライブラリをインポートする

使用する IVI-COM 計測器ドライバのタイプライブラリをインポートします。

1 **Visual Basic で［ツール］＞［参照設定］をクリックします。**

参照設定ダイアログが表示されます。

2 **［IviDCPwr 2.0 TypeLibrary］、［IviDriver 1.0 Type Library］、［IviSessionFactory 1.1 Type Library］をチェックします。**



3 **［OK］をクリックします。**

プログラムを構成する

オブジェクトの作成とセッションのオープン／クローズ

計測器ドライバオブジェクトの作成と、セッションをオープン／クローズするコードを記述します。「新規のマクロを作成する」（p.7）で開いたボタンハンドラのソースコードに、下記のサンプルコードを記述します。

```
Sub CommandButton1_Click()

      Dim sf As IIviSessionFactory
      Set sf = New IviSessionFactory

      Dim inst As IIviDCPwr
      Set inst = sf.CreateDriver("mySupply")

      inst.Initialize "mySupply", True, True, ""

      inst.Close

End Sub
```

サンプルコードの解説

クラス・インターフェースでは、特定の測定器に依存するコードを記述しません。そのため、ソースコード上に Kikusui という名前は使いません。IviSessionFactory オブジェクトを作成し、IIviSessionFactory インターフェースへの参照を取得します。

次に、「Logical Name を設定する」（p.20）で作成した IVI ロジカルネーム（仮想インストルメント）「mySupply」を指定して CreateDriver メソッドを呼び出します。

最後に、Initialize メソッドを呼び出して IVI ロジカルネームを渡し、セッションをクローズします。IVI ロジカルネームを渡すと、プログラム実行時には「Hardware Asset を作成する」（p.17）で指定した VISA アドレスが適用されます。この時点で、計測器との通信が開始されます。

上記の記述だけでも、プログラムを実行することが可能です。動作確認のためにプログラムを実行する場合には、「プログラムを実行する」（p.27）を参照してください。

Initialize メソッドのパラメータ

すべての IVI-COM 計測器ドライバには、IVI 仕様書で定義された Initialize メソッドがあります。Initialize メソッドには、下記のパラメータがあります。

パラメータ	タイプ	説明
ResourceName	String	計測器を指定する文字列です。クラス・インターフェースでは、IVI ロジカルネームで指定します。
IdQuery	Boolean	TRUE を指定した場合、計測器に対して "*IDN?" クエリなどを発行して機種情報を問い合わせます。
Reset	Boolean	TRUE を指定した場合、"*RST" コマンドなどを発行して計測器の設定をリセットします。
OptionString	String	IVI 定義の設定を変更できます。

OptionString を設定するには

OptionString では、下記の IVI 定義を設定できます。

IVI 定義	デフォルト値
RangeCheck	TRUE
Cache	TRUE
Simulate	FALSE
QueryInstrStatus	FALSE
RecordCoercions	FALSE
Interchange Check	FALSE

OptionString は文字列のパラメータです。下記に、サンプルを記載します。

```
QueryInstrStatus = TRUE , Cache = TRUE , DriverSetup=12345
```

書式について、次の点に注意してください。

- 設定値を指定しない場合にはデフォルト値が適用されます。
- 設定値は Boolean 型です。「TRUE」、「FALSE」、「1」、「0」のいずれかが設定できます。
- 機能名称および設定値はケース・インセンシティブです。大文字と小文字は区別されません。
- 複数の項目を設定する場合には、カンマで区切ります。
- 計測器ドライバによっては、「DriverSetup」パラメータがサポートされています。
  「DriverSetup」は、IVI 仕様書では定義されない項目を Initialize の呼び出し時に指定するパラメータです。利用目的や書式はドライバに依存します。そのため、「DriverSetup」を設定する場合には、「Option String」の最後の項目として指定します。「DriverSetup」の指定内容については、ドライバの Readme またはオンライン・ヘルプなどを参照してください。

チャンネルと試験条件を設定する

電源装置などの場合、IVI 計測器ドライバは複数のチャンネルが装備されていることを前提に設計されています。そのため、計測器のパネル設定について操作をするプロパティやメソッドは、IVI 仕様書でリピーテッド・キャパビリティ（一般的な COM の用語ではコレクション）と呼ばれるオブジェクト配列の概念を実装するケースが多く見られます。直流電源の計測器ドライバの場合には、Output コレクションが該当します。

クラス・インターフェースの場合には、IIviDCPwrOutput を使用します。下記は、「プログラムを構成する」（p.22）のコードに計測器の出力チャンネル制御を追加したサンプルコードです。

```
Sub CommandButton1_Click()
      Dim sf As IIviSessionFactory
      Set sf = New IviSessionFactory

      Dim inst As IIviDCPwr
      Set inst = sf.CreateDriver("mySupply")

      inst.Initialize "mySupply", True, True, ""

      Dim output As IIviDCPwrOutput
      Set output = inst.outputs.Item("Track_A")

      output.VoltageLevel = 20.0
      output.CurrentLimit = 2.0
      output.Enabled = True

      inst.Close
End Sub
```

サンプルコードの解説

IviDCPwr インターフェースの Outputs プロパティを通じて IIviDCPwrOutputs を取得します。Item プロパティでは、「Virtual Name を作成する」（p.19）で作成したバーチャルネーム「Track_A」を渡します。バーチャルネームを渡すと、プログラム実行時には Physical Name で指定したチャンネル名が適用されます。

VoltageLevel プロパティは、電圧レベルを設定します。CurrentLimit プロパティは、電流リミットを設定します。Enabled プロパティは、出力の ON ／ OFF を設定します。

エラーを処理する

範囲外の値をプロパティに渡したり、サポートされていない機能を呼び出したりすると、計測器ドライバにエラーが発生します。IVI-COM 計測器ドライバでは、計測器ドライバ内で発生したエラーは COM 例外としてクライアントプログラムに伝えられます。VBA の場合には、COM 例外は On Error Goto ステートメントを使って処理できます。

下記は、「チャンネルと試験条件を設定する」（p.25）のコードにエラー処理を加えたサンプルコードです。

```
Sub CommandButton1_Click()

      On Error GoTo DRIVER_ERR:

      Dim sf As IIviSessionFactory
      Set sf = New IviSessionFactory

      Dim inst As IIviDCPwr
      Set inst = sf.CreateDriver("mySupply")

      inst.Initialize "mySupply", True, True, ""

      Dim output As IIviDCPwrOutput
      Set output = inst.outputs.Item("Track_A")

      output.VoltageLevel = 20.0
      output.CurrentLimit = 2.0
      output.Enabled = True

      inst.Close

      Exit Sub

DRIVER_ERR:
      Debug.Print Err.Description

End Sub
```

サンプルコードの解説

このコードでは、On Error Goto ステートメントを使ってエラーを処理しています。たとえば、Item プロパティに渡した名前が間違っている場合、VoltageLevel に設定する値が適正範囲から外れている場合、または計測器との通信に失敗した場合には、いずれも計測器ドライバ内で COM 例外が発生します。上記のコードでは、例外が発生した場合に簡単なメッセージをイミディエイトウインドウに表示します。

プログラムを実行する

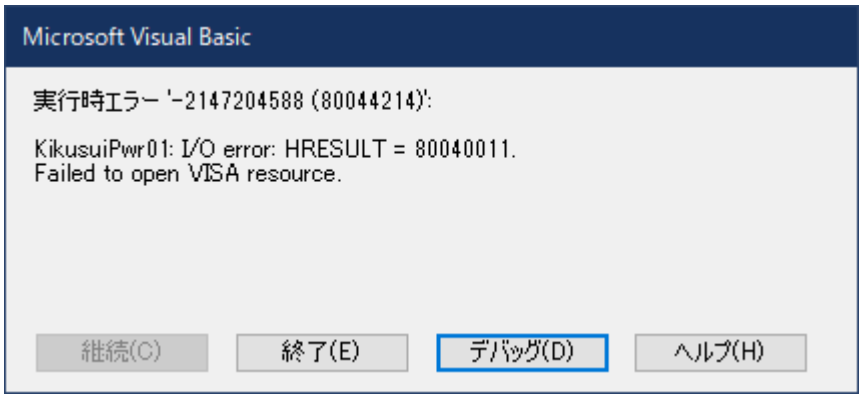
構成したプログラムを実行します。

1 **Visual Basic のワークシート上に作成したボタンをクリックします。**

VBA マクロが動作し、プログラムが終了します。

実行に失敗した場合

COM 例外（VBA ランタイム・エラー）が発生します（p.26）。



計測器を交換する

計測器を交換した場合には、仮想インストルメント（IVI コンフィグレーション）の Driver Session を変更するだけで、動作を継続できます。アプリケーション自体を変更する必要はありません。

変更する Driver Session の設定は、以下の 3 つです。

項目	設定内容
［Hardware］タブ＞［Hardware Assets］＞［Resource Descriptor］	計測器の接続先 VISA アドレス
［Software］タブ＞［Software Module］	使用する計測器ドライバ
［Virtual Names］タブ＞［Physical Names］	仮想チャンネル名のマップ先物理名

交換した計測器に合わせて正しく設定すれば、アプリケーションを再度コンパイル・リンクすることなく動作します。

本ガイドの例では、計測器を Kikusui PWR-01 シリーズ直流電源（kipwr 計測器ドライバでホストされる計測器）から Agilent N5700 シリーズ直流電源（AgN57xx ドライバでホストされる計測器）に交換した場合、［mySupply］の設定を以下のように変更します。

項目	変更内容
［Hardware］タブ＞［Hardware Assets］＞［Resource Descriptor］	Kikusui PWR-01 シリーズ直流電源の接続先 VISA アドレス ⇒ Agilent N5700 シリーズ直流電源の接続先 VISA アドレス
［Software］タブ＞［Software Module］	「kipwr」⇒「AgN57xx」
［Virtual Names］タブ＞［Physical Names］	「OutputChannel1」⇒「Output1」

Memo

IVI クラス・ドライバを利用したインターチェンジャビリティ機能は、計測器の交換前後での動作を保証するものではありません。交換後のシステムが正常に機能するかどうかを十分に検証してから運用してください。