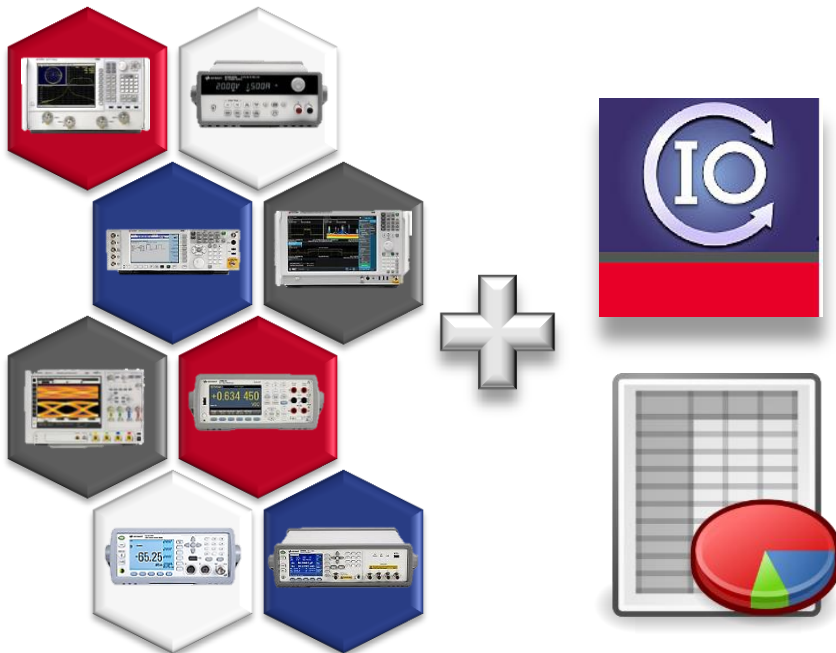


Keysight Technologies IO Libraries Suite Excel / VBA による 測定器制御プログラム入門 中級編

- VISA COM ライブラリの使い方
- 測定器制御のテクニック
- Excel / VBA のテクニック



Application Note

目次

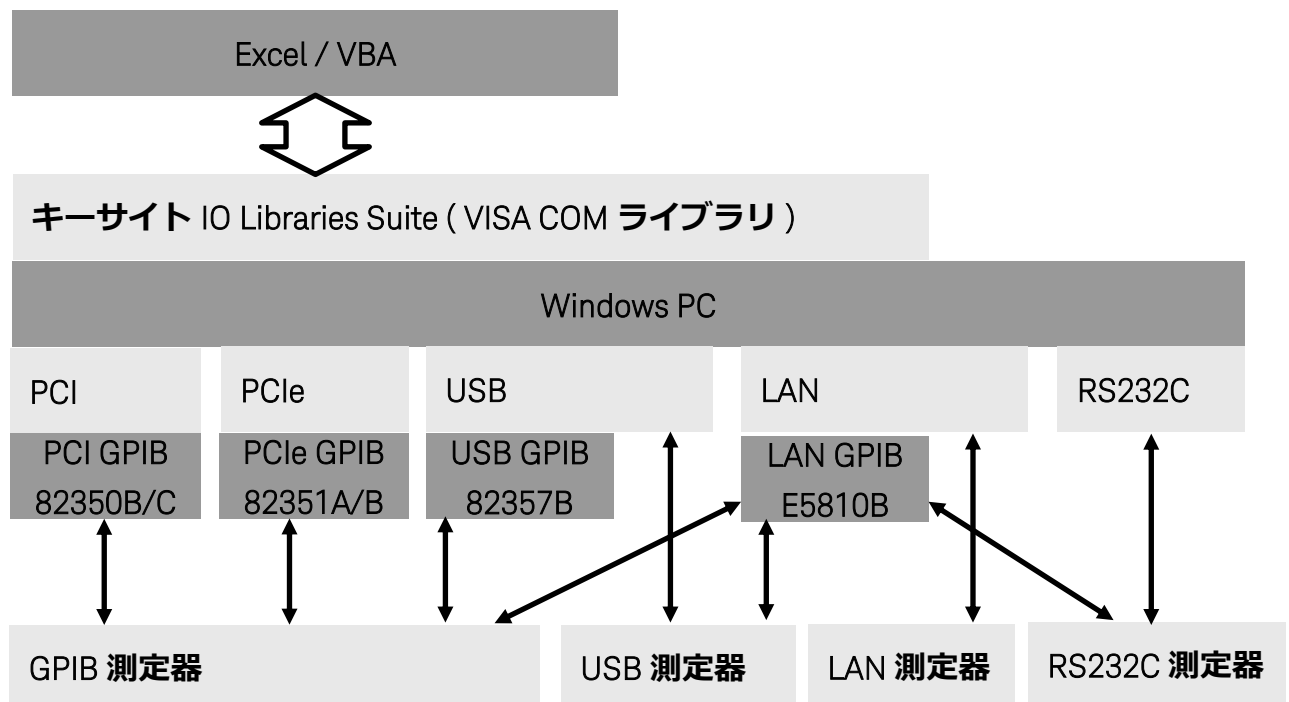
1. はじめに	p. 3
1) キーサイト IO Libraries Suite のインストール	p. 4
2) Excel / VBA の設定	P. 4
2. VISA COM ライブラリの使い方	p. 5
1) 概要	p. 5
2) 測定器のオープン・クローズと文字列の入出力	p. 6
3) 数値データの入出力	p. 7
4) 複数の文字列や数値の入出力	p. 8
5) VISA COM ライブラリの代表的なエラー	p. 9
3. 測定器制御のテクニック	p. 11
1) 測定器のエラーを調べる	p. 11
2) 測定器のステータスを調べる	p. 12
3) 測定器のリモート/ローカル制御	p. 13
4. Excel / VBA のテクニック	p. 15
1) プログラムの実行に待ち時間を入れる	p. 15
2) 測定器の応答文字列の最後から<LF>を取り除く	p. 15
3) プログラムの経過時間を調べる	p. 16
4) 整数型変数のビットを判断する	p. 16
5) Excel / VBA のエラー処理	p. 17
5. サンプル・プログラム	p. 18
1) 測定器のメモリ上の測定値を Excel シートのセルに転送する	p. 18
2) 測定を行い測定値を Excel シートのセルに転送する	p. 19
3) 複数の測定器を制御する	p. 20
6. VISA COM ライブラリ 簡易リファレンス	p. 21
7. 参考資料	p. 26

1. はじめに

- 測定器を GPIB / LAN / USB などのインタフェースで PC と接続することで、測定器のリモート制御が可能です。
- Excel / VBA でプログラムを作成することにより、自動計測が可能です。

最近の測定器には、インタフェースとして、GPIB だけではなく LAN や USB が装備されている機種があります。LAN や USB を使用すると、PC にインタフェースを追加することなく、測定器を PC に接続できます。GPIB を使用して測定器を接続する場合、PC に GPIB インタフェースを追加することで、測定器と接続できます。

Excel / VBA からキーサイト IO Libraries Suite により提供される VISA COM ライブラリを呼び出すプログラムを作成することにより、PC に接続された測定器を制御する「自動計測」を行うことが可能です。



(GPIB 接続の測定器を制御する場合、キーサイトの GPIB インタフェースをご利用ください。)

このアプリケーション・ノートでは、PC と測定器が接続されており、キーサイト IO Libraries Suite から認識できている環境を前提に、Excel / VBA から IO Libraries Suite に含まれている VISA COM ライブラリを呼び出して、測定器を制御する方法を説明します。

さらに、測定器制御のテクニックや、測定器制御に必要な Excel / VBA のテクニックなどもご紹介いたします。

Excel / VBA の一般的なプログラム方法などは、Excel の Help や市販の書籍などをご利用ください。

測定器のコマンドについては、測定器のマニュアルをご参照ください。

VISA COM ライブラリの詳細は、IO Libraries Suite に含まれる Help ファイルをご参照ください。

このアプリケーション・ノートのサンプル・プログラムは、キーサイト製 デジタルマルチメータ 34465A で動作を確認しています。サンプル・プログラム中で測定器に出力するコマンドの部分をお使いの測定器に合わせて修正することにより、他の測定器にも応用できます。

1) キーサイト IO Libraries Suite のインストール

IO Libraries Suite は以下の URL よりダウンロードが可能です。

トライアル & ライセンスより [詳細 & ダウンロード] をクリックしてください。

(過去のバージョンもダウンロードできます。)

[IO Libraries Suite 最新版]

<http://www.keysight.com/find/iolib>

[IO Libraries Suite 最新版 ダウンロードページ]

<http://www.keysight.com/find/iosuitedownload>

詳細は下記 Readme または Help をご参照ください。

[IO Libraries Suite 17.1 Readme]

http://www.keysight.com/upload/cmc_upload/All/readme_Keysight_IO_Libraries_Suite_17_1.htm

IO Libraries Suite の最新の日本語の簡易取扱説明書は下記をご参照ください。

[IO Libraries Suite 簡易取扱説明書 最新版]

http://www.keysight.co.jp/find/iolib_jp_quick

IO Libraries Suite 簡易取扱説明書 などをご参考に、PC と測定器の接続を確認してください。

2) Excel / VBA の設定

このアプリケーション・ノートで紹介するサンプル・プログラムは、Excel のシートにボタンを配置し、そのボタンのクリック・イベントハンドラとして作成しています。Excel のシートにボタンを配置するには、Excel に開発タブを表示させると便利です。また、Excel / VBA のイベントハンドラから VISA COM ライブラリを呼び出すには、Excel の Visual Basic for Application のメニューから、ツール > 参照設定 で、VISA COM x.x Type Library を参照に追加する必要があります。

Excel に開発タブを表示させる方法や VISA COM x.x Type Library を参照に追加する方法、ボタンのクリック・イベントハンドラを作成する方法などは、以下の資料をご参照ください。

[Excel 2010 / VBA による測定器制御入門 VISA COM ライブラリ編]

<http://literature.cdn.keysight.com/litweb/pdf/5991-2363JAJP.pdf>

ご利用の IO Libraries Suite のバージョンにより、VISA COM Type Library のバージョンが異なります。この資料で紹介する内容は、VISA COM Type Library のバージョンが、1.0、3.0、5.2、5.5 のどのバージョンでも同様に動作します。2015 年 6 月時点での最新の IO Libraries Suite 17.1 では、VISA COM 5.5 Type Library です。

2. VISA COM ライブラリの使い方

1) 概要

VISA COM ライブラリで測定器を制御するには、以下の処理を行います。

- 測定器のアドレスを指定して測定器をオープンする
- オープンした測定器にコマンドを出力し、応答を受け取る
- 測定器をクローズする

プログラムは、測定器にコマンドを送る前に、一度だけ、測定器をオープンします。

プログラムは、測定器のオープンに成功すると、その測定器にコマンドを送ったり、応答を受け取ったりすることができます。

測定器にコマンドを送り、応答を受け取る処理は、必要に応じて繰り返します。

プログラムは、これ以上測定器を使用しない場合、測定器をクローズします。

【VISA アドレス】

VISA COM ライブラリでは、測定器をオープンするときに、測定器のアドレスとして VISA アドレスを使用します。VISA アドレスは、インタフェースを特定する部分と、測定器を特定する部分から構成されています。

VISA アドレスの例)

GPIB0::22::INSTR

PC に接続されている1台目の GPIB インタフェース (GPIB0) に接続されている、GPIB アドレス 22 番の機器

TCPIP0::169.254.4.10::inst0::INSTR

LAN に接続されている IP アドレス 169.254.4.10 の測定器

TCPIP0::169.254.4.10::5025::SOCKET

LAN に接続されている IP アドレス 169.254.4.10 の測定器。TCP ポート番号 5025 で、ソケットプロトコルで接続

USB0::0x0957::0x0A07::MY00000123::0::INSTR

USB に接続されている、ベンダ ID 0x0957、プロダクト ID 0x0A07、シリアル番号 MY00000123 の測定器

VISA COM ライブラリを使用するプログラムは、主に、以下の 3 種類のオブジェクトを使用して、測定器の制御を行います。

- ResourceManager オブジェクト
- IO Resource オブジェクト
- FormattedIO488 オブジェクト

ResourceManager オブジェクトは、プログラムで作成します。ResourceManager オブジェクトの主な役割は、IO Resource オブジェクトを作成することです。プログラムは、ResourceManager オブジェクトの Open メソッドに VISA アドレスを渡して、IO Resource オブジェクトを作成します。

IO Resource オブジェクトは、ResourceManager オブジェクトにより作成されます。測定器との実際の通信を行うオブジェクトです。

FormattedIO488 オブジェクトは、プログラムで作成します。通常は IO プロパティに、ResourceManager オブジェクトにより作成された IO Resource オブジェクトを保存します。

FormattedIO488 オブジェクトはプログラムにより呼び出され、入出力データのフォーマット変換を行います。

FormattedIO488 オブジェクトは IO Resource オブジェクトを呼び出し、実際の通信を IO Resource オブジェクトに依頼します。

ResourceManager オブジェクトは、プログラムにつき 1 個、IO Resource オブジェクトと FormattedIO488 オブジェクトは、測定器ごとに 1 個、作成します。

2) 測定器のオープン・クローズと文字列の入出力

【サンプル・プログラム】

```

Sub ボタン1_Click()
    Dim RM As New VisaComLib.ResourceManager ' ResourceManager オブジェクトの宣言と作成
    Dim DMM As New VisaComLib.FormattedIO488 ' FormattedIO488 オブジェクトの宣言と作成
    Set DMM.IO = RM.Open("GPIB0::22::INSTR") ' IO Resource オブジェクトの作成

    DMM.WriteString "*IDN?" ' 測定器にコマンドを出力
    Dim strIDN As String ' 測定器からの応答を保存する変数の宣言
    strIDN = DMM.ReadString ' 測定器から応答を入力

    DMM.IO.Close ' IO Resource オブジェクトの解放
    Set DMM = Nothing ' FormattedIO488 オブジェクトの解放
    Set RM = Nothing ' ResourceManager オブジェクトの解放
End Sub

```

このプログラムは、最初に VISA COM のオブジェクトを作成し、測定器をオープンしています。次に、測定器に *IDN? コマンドを出力し、測定器からの応答を受け取り、最後に測定器をクローズし、VISA COM のオブジェクトを解放しています。

- 1 ResourceManager オブジェクトと FormattedIO488 オブジェクトの宣言と作成を行っています。
- 2 ResourceManager オブジェクトの Open メソッドで作成した IO Resource オブジェクトを、1 で作成した FormattedIO488 オブジェクトの IO プロパティに代入しています。
- 3 WriteString メソッドは、引数として文字列データを受け取り、測定器に出力します。ここでは、測定器に *IDN? という文字列を、コマンドとして出力しています。
- 4 ReadString メソッドは、測定器からの応答を受け取り、戻り値として文字列データを返します。ここでは、測定器から、*IDN? コマンドの応答を受け取り、String 型の変数 strIDN に代入しています。
- 5 IO.Close で、IO Resource オブジェクトの Close メソッドを実行し、測定器をクローズしています。
- 6 FormattedIO488 オブジェクトと ResourceManager オブジェクトを解放しています。

通常のプログラムでは、1 と 2 の部分は、測定器の制御を開始する前に一度だけ実行します。その後、3 と 4 を必要に応じて実行します。5 と 6 は、プログラムでは、これ以上測定器を使用しないときに実行します。

このサンプル・プログラムは短いプログラムですが、測定器へのコマンド出力と、測定器からの応答の入力を行っていますので、コマンドを変更することで、多くの測定に応用することが可能です。

SCPI (Standard Commands for Programmable Instruments) コマンドをサポートしている測定器は、*IDN? コマンドを受け取ると、メーカー名、型番などの情報を返します。*IDN? コマンドは、測定器との通信の確認に利用できます。

3) 数値データの入出力

【サンプル・プログラム】

```

Sub ボタン1_Click()
    Dim RM As New VisaComLib.ResourceManager
    Dim DMM As New VisaComLib.FormattedIO488
    Set DMM.IO = RM.Open("GPIB0::22::INSTR")

    DMM.WriteString "CONF:VOLT:DC ", False ' 出力バッファに"CONF:VOLT:DC "をコピー } 1

    Dim dblRange As Double
    dblRange = 10
    DMM.WriteNumber dblRange, ASCIIType_R8 ' 出力バッファに変数 dblRange の値を追加し、出力 } 2

    DMM.WriteString "READ?" ' READ? を出力
    Dim dblData As Double
    dblData = DMM.ReadNumber(ASCIIType_R8) ' 応答を変数 dblData に保存

    DMM.IO.Close } 3
    Set DMM = Nothing
    Set RM = Nothing
End Sub

```

このプログラムは、測定器に CONF:VOLT:DC 10 コマンドと READ? コマンドを出力し、測定器からの応答を Double 型の変数に読み込んでいます。

FormattedIO488 オブジェクトの WriteNumber メソッドを使用すると、Excel / VBA の Double 型などの数値データを文字列に変換し、測定器に出力することができます。ReadNumber メソッドを使用すると、測定器からの数値データの応答を、Excel / VBA の Double 型などの変数に読み込むことが可能です。

1 WriteString メソッドは、第 2 引数に何も指定しない、あるいは、True を指定すると、第 1 引数で指定された文字列データを測定器に出力します。ここでの WriteString メソッドは、第 2 引数に False を指定しています。第 2 引数に False を指定すると、WriteString メソッドは、第 1 引数の文字列を出力バッファにコピーするだけで、測定器には出力されません。ここでは、出力バッファに、CONF:VOLT DC がコピーされます。

2 WriteNumber メソッドは、第 1 引数で渡されたデータを、第 2 引数で指定されたフォーマットを使用して文字列に変換し、出力バッファを経由して測定器に出力します。**2** では、10 という数値データの出力しか指定していませんが、出力バッファには、すでに CONF:VOLT:DC がコピーされていますので、測定器には、CONF:VOLT:DC 10 が出力されます。

3 DMM.ReadNumber(ASCIIType_R8) では、測定器からの応答を、倍精度の実数として解釈し、Double 型の変数に代入しています。

ReadNumber メソッドは引数で指定したタイプにかかわらず、戻り値として常に Double 型を返します。例えば、

```

Dim resp As Integer
resp = DMM.ReadNumber(ASCIIType_I2)

```

を実行すると、DMM.ReadNumber(ASCIIType_I2) は、Double 型のデータを返しますが、代入している変数 resp が Integer 型のため、Excel / VBA が Integer 型に変換して、変数 resp に代入します。

4) 複数の文字列や数値の入出力

【サンプル・プログラム】

```

Sub ボタン1_Click()
    Dim RM As New VisaComLib.ResourceManager
    Dim DMM As New VisaComLib.FormattedIO488
    Set DMM.IO = RM.Open("GPIB0::22::INSTR")

    DMM.WriteString "*IDN?"
    Dim strData() As String
    strData = DMM.ReadList(ASCIIType_BSTR) ' カンマ区切りで応答の文字列を String 型の配列に読み込む } 1

    DMM.WriteString "SAMP:COUN 10;:READ?" ' 応答は10個の数値がカンマ区切りで返されます
    Dim dblData() As Double
    dblData = DMM.ReadList(ASCIIType_R8) ' カンマ区切りで応答の数値を Double 型の配列に読み込む } 2

    DMM.IO.Close
    Set DMM = Nothing
    Set RM = Nothing
End Sub

```

このプログラムは、*IDN? コマンドの応答をカンマで区切り、String 型の配列に読み込みます。
次に、SAMP:COUN 10;:READ? コマンドを出力し、応答を Double 型の配列に読み込みます。

FormattedIO488 オブジェクトの ReadList メソッドを使用すると、測定器から返される複数の応答を VBA の指定した型の変数の配列に読み込むことができます。

例えば、測定器からカンマ区切りで複数の文字列が返される場合、String 型の配列の変数に、一度に読み込むことができます。
また、測定器からカンマ区切りで複数の数値が返される場合、Double 型の配列の変数に、一度に読み込むことができます。

FormattedIO488 オブジェクトの WriteList メソッドを使用すると、VBA の配列の変数の内容を、ASCII に変換して測定器に出力します。測定器に出力されるデータは、カンマで区切られます。

1 測定器からの応答をカンマで区切り、String 型の配列として返します。

2 測定器からの応答をカンマで区切り、Double 型の配列として返します。

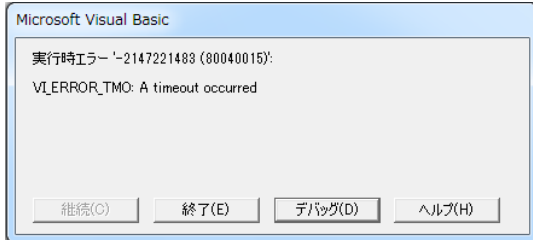
WriteList メソッドの使用例は 6. VISA COM ライブラリ 簡易リファレンス P.25 をご参照ください。

5) VISA COM ライブラリの代表的なエラー

Excel / VBA から VISA COM ライブラリを呼び出したときに、VISA COM ライブラリの内部でエラーが発生すると、Excel / VBA はデフォルトの動作として、エラーのダイアログを表示させ、プログラムの実行を中断します。

ここでは、VISA COM ライブラリが発生させる代表的なエラーについて、原因、対処方法を紹介します。

【測定器との通信中のエラー（タイムアウトエラー）】



- 原因

このエラーは、ReadString や WriteString などのメソッドを実行したときに、測定器からの応答があらかじめ決められた時間（タイムアウト時間）内に返らない場合に発生します。以下のような原因が考えられます。

- 1) 測定器と PC の接続が切れてしまった場合
- 2) 測定器が応答を返すコマンドを受け取っていないときに、応答を読み出した場合
- 3) 測定器が応答を返すまでに、タイムアウト時間以上に時間がかかる場合
- 4) ソケットプロトコル（LAN接続）で、終端文字で読み込みを完了させる設定を行っていない場合

- 対策 1)

測定器との接続を確認します。IO Libraries Suite のツール Interactive IO で接続の確認が行えます。

- 対策 2)

多くの測定器は、応答を返すコマンドを受け取らない限り応答を返しません。例えば、*IDN? コマンドを測定器に送るときに、誤って *IDN と送ってしまった場合、測定器は、解釈できないコマンドが送られたためにエラーが発生します。34465A の場合は、-113, “Undefined header” というエラーが発生します。測定器はコマンドを実行できないので応答は返さず、プログラムで応答を読み取ると、タイムアウトエラーが発生します。

この場合は、測定器から応答を受け取る前に、測定器に送っているコマンドを確認してください。

- 対策 3)

VISA COM ライブラリのデフォルトのタイムアウト時間は 2 秒です。測定器から応答が返るまでに 2 秒以上かかる場合、タイムアウト時間をあらかじめ、適切な値に設定します。

```
DMM.IO.Timeout = 10000
```

上記は、タイムアウト時間を 10 秒に設定する例です。

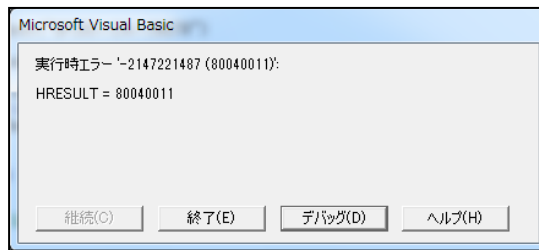
- 対策 4)

LAN 接続の測定器をソケットプロトコルで接続したときに、終端文字で読み込みを終了させる設定を行っていない場合にも発生します。LAN の測定器をソケットプロトコルで接続した場合、終端文字で読み込みを終了させる設定を行ってください。

```
Set DMM.IO = RM.Open("TCPIP0::169.254.4.10::5025::SOCKET")
DMM.IO.TerminationCharacter = 10 ' データの終端を示す文字は、ASCII の 10、ラインフィード <LF>
DMM.IO.TerminationCharacterEnabled = True ' 終端文字を受け取ると、読み取りを終了する
```

上記は、終端文字としてラインフィードキャラクタを指定している例です。

【 測定器をオープンする時のエラー 】



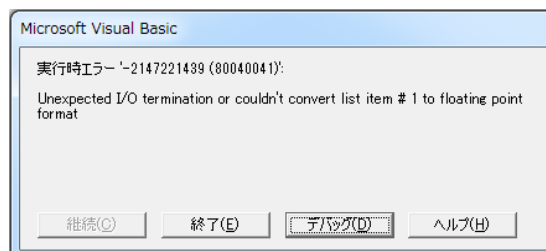
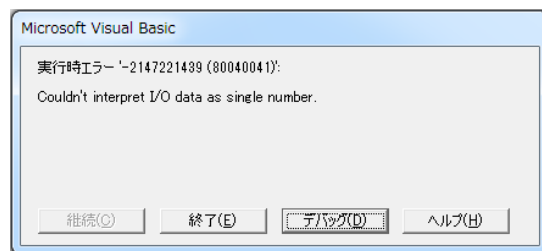
- 原因

このエラーは、ResourceManager オブジェクトの Open メソッドを呼び出したときに発生します。Open メソッドでは、引数として渡された VISA アドレスの測定器に対してオープンを試みますが、測定器の接続が確認できない場合にこのエラーが発生します。

- 対策

Open メソッドに指定した VISA アドレスが正しいかどうか、また、VISA アドレスで指定している測定器が PC と接続されているかどうか、を確認します。

【 ReadNumber、ReadList のエラー 】



- 原因

このエラーは、FormattedIO488 オブジェクトの ReadNumber メソッド、ReadList メソッドで数値のフォーマットを指定して呼び出したときに、測定器からの応答が数字として解釈することができない場合に、発生します。

また、測定器からの応答が、タイムアウト時間内に得られない場合も含まれます。

- 対策 1)

測定器からの応答が、数字として解釈できる内容であることを確認してください。確認のため、一時的に、ReadNumber メソッドを ReadString メソッドに変更し、文字列として受け取り、応答の内容を文字列として確認する方法があります。

また、IO Libraries Suite のツール IO Monitor を使用して、測定器からの応答を確認する方法もあります。

- 対策 2)

タイムアウト時間内に応答が返らない場合は、前項の「測定器との通信中のエラー（タイムアウトエラー）」をご参照ください。

3. 測定器制御のテクニック

1) 測定器のエラーを調べる

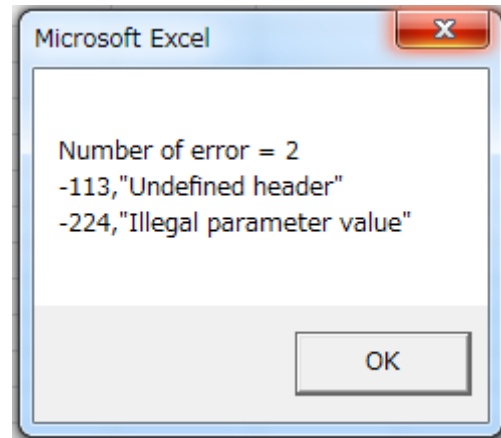
測定器は、測定器で発生したエラーを順番にメモリに保存しており、PC からの問い合わせに対して、発生したエラーの情報を返すことができます。

リモートで測定器を制御する場合、「測定器の設定が終了し、測定を開始する前」、あるいは、「測定完了後」など、適切なタイミングで測定器にエラーを問い合わせをことをお勧めします。これにより、測定器が期待どおりに動作していることが確認できます。

測定器にエラー情報を問い合わせるには、SYST:ERR? コマンドを使用します。測定器にエラーが発生していない場合、測定器は、+0,"No error" を返します。測定器から、+0,"No error" 以外のメッセージが返された場合、複数のエラーが発生している可能性がありますので、+0,"No error" が返ってくるまで、繰り返し、SYST:ERR? コマンドを送り、応答を受け取ります。

```
DMM.WriteString "ABC" ' ABC というコマンドはないのでエラー
DMM.WriteString "MEAS:VOLT:DC? A" ' A というレンジはないのでエラー
```

```
Dim strErr As String, strAllErr As String, intErrNum As Integer
strAllErr = ""
intErrNum = 0
Do
    DMM.WriteString "SYST:ERR?"
    strErr = DMM.ReadString
    If strErr = "+0,\"No error\"" & vbLf Then
        Exit Do
    End If
    strAllErr = strAllErr & strErr
    intErrNum = intErrNum + 1
Loop
MsgBox "Number of error = " & intErrNum & vbCrLf & strAllErr
```



1 ここでは、無限ループの中で、SYST:ERR? コマンドを測定器に送り、応答を受け取っています。測定器からの応答が、+0,"No error" であれば、無限ループを抜けますが、それ以外の場合は、測定器にエラーメッセージが残っている可能性がありますので、+0,"No error" が返るまで、SYST:ERR? を送り、応答を受け取っています。

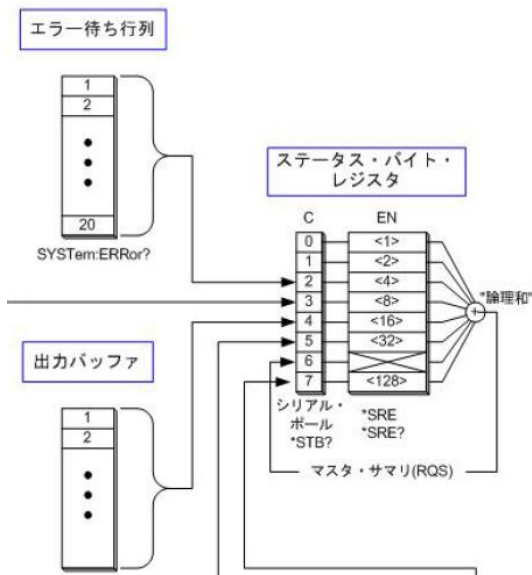
測定器のエラー番号とエラーメッセージは、測定器ごとに異なりますので、エラー番号、エラーメッセージの詳細は測定器のマニュアルでご確認ください。

2) 測定器のステータスを調べる

測定器は、“エラーの有無”、“トリガ待ち”など、測定器自体のステータスを保持しており、プログラムからの問い合わせにステータスを返すことができます。

プログラムは、測定器からステータス情報を読み取ることで、測定器が期待通りのステータスになっていることを確認することができます。

以下の図は、34465A の「ステータス・バイト・レジスタ」（ステータスシステムの一部）の例です。



この図からは、34465A の「ステータス・バイト・レジスタ」の Bit2 は、「エラー待ち行列」の内容が反映していること、また、Bit4 は、「出力バッファ」の内容が反映していることがわかります。

「ステータス・バイト・レジスタ」は、*STB? コマンドで読み取ることができます。

```
DMM.WriteString "RST" ' RST コマンドは誤りのため、エラーが発生します。正しくは *RST
' 34465A の場合、エラーが発生するとステータスバイトレジスタの Bit2 が 1 になる
Dim resp As Integer
DMM.WriteString "*STB?" ' ステータスバイトの要求
resp = DMM.ReadNumber(ASCIIType_I2) ' ステータスバイトの内容を、2 バイト整数に読み込む
' Bit2を確認するには、4 で割って (2 ビット右にシフトして) 奇数か偶数か (Bit0 が 1 かどうか)を確認する
If ((resp \ 4) Mod 2) = 1 Then
    Dim strErr As String
    DMM.WriteString "SYS:ERR?"
    strErr = DMM.ReadString
    MsgBox "Error " & vbCrLf & strErr
Else
    MsgBox "No error"
End If
```

Excel / VBA で、\ は、除算の商、Mod は除算の余り、を計算する演算子です。上記の例では、\ 4 を行い、4 で割った整数部分を取るにより、2 ビット右にシフトしています。さらに、結果に Mod 2 を行うことで、2 で割った余り（奇数か偶数か）を判断しています。

Status Byte Register、あるいは、その他のステータスを示すレジスタの内容は測定器ごとに異なりますので、詳細は各測定器のマニュアルをご参照ください。

3) 測定器のリモート/ローカル制御

多くの測定器はリモート状態とローカル状態を持っており、インタフェースからコマンドを受け取るとリモート状態になります。測定器がリモートの状態のとき、測定器のフロントパネルのキーは、Local キー 以外は操作できなくなります。これにより、リモート制御中にユーザの誤操作を防止します。

リモート状態の測定器は、Local キーを押すことによりローカル状態にもどり、通常のフロントパネル操作が可能になります。

リモート/ローカルの測定器の状態は、通常は測定器のコマンドにより制御するものではなく、測定器と接続しているインタフェースの機能により実現します。

【 GPIB 接続の場合】

GPIB インタフェースには、REN という信号線があり、REN の信号線の状態と、GPIB のバスコマンドにより、測定器のリモート/ローカルを制御しています。

```
Dim RM As New VisaComLib.ResourceManager
Dim DMM As New VisaComLib.FormattedIO488
Set DMM.IO = RM.Open("GPIB0::22::INSTR") ' GPIB::INSTR タイプの IO Resource オブジェクトを作成
DMM.WriteString "*RST" ' ここで測定器はリモートになる

Dim igpibDev1 As VisaComLib.IGpib ' GPIBの機能を使用する IO Resource オブジェクトのインタフェース
Set igpibDev1 = DMM.IO ' 既存の IO Resource オブジェクトを代入し、GPIB の機能を使用する

igpibDev1.ControlREN GPIB_REN_GTL ' この行を実行すると測定器はローカルになる

DMM.WriteString "CLS" ' この行を実行すると、再び、測定器はリモートになる

Set igpibDev1 = Nothing
DMM.IO.Close
Set DMM = Nothing
Set RM = Nothing
```

1 GPIB インタフェース特有の機能の使用するために、IO Resource オブジェクトの IGpib インタフェースの変数 (igpibDev1) を使用しています。プログラムからは、igpibDev1 を使用することで、GPIB 接続の測定器に対して、リモート/ローカルを制御することができます。

2 IGpib インタフェースの ControlREN メソッドを、GPIB_REN_GTL を指定して呼び出すことにより、GPIB 接続の測定器をローカル状態にしています。

【 USB 接続の場合 】

USB インタフェースには GPIB の REN に相当する信号線はありません。USB のプロトコルにより、測定器のローカル/リモート制御を実現しています。

```
Dim RM As New VisaComLib.ResourceManager
Dim DMM As New VisaComLib.FormattedIO488
Set DMM.IO = RM.Open("USB0::0x2A8D::0x0101::MY54500172::0::INSTR")
DMM.WriteString "*RST" ' ここで測定器はリモートになる
Dim iusbDev1 As VisaComLib.IUsb ' USB の機能を使用するためのインタフェース
Set iusbDev1 = DMM.IO

iusbDev1.ControlREN GPIB_REN_GTL ' ここで測定器はローカルになる
```

} 1
} 2

1 USB インタフェース特有の機能の使用するために、IO Resource オブジェクトの IUsb インタフェースの変数 (iusbDev1) を準備しています。プログラムからは、iusbDev1 を使用することで、USB 接続の測定器に対して、リモート/ローカルを制御することができます。

2 IUsb インタフェースの ControlREN メソッドを、GPIB_REN_GTL を指定して呼び出すことにより、USB 接続の測定器をローカル状態にしています。

【 LAN 接続 (HiSLIP プロトコル) の場合 】

LAN インタフェースには GPIB の REN に相当する信号線はありませんが、HiSLIP プロトコルで接続した場合は、プロトコルの機能で、リモート/ローカルの制御が可能です。

```
Dim RM As New VisaComLib.ResourceManager
Dim DMM As New VisaComLib.FormattedIO488
Set DMM.IO = RM.Open("TCPIP0::169.254.4.70::hislip0::INSTR") ' HiSLIP 用の VISA アドレス
DMM.WriteString "*RST" ' ここで測定器はリモートになる
Dim ihislipDev1 As VisaComLib.IHislipInstr ' HiSLIP の機能を使用するためのインタフェース
Set ihislipDev1 = DMM.IO

ihislipDev1.ControlREN GPIB_REN_GTL ' ここで測定器はローカルになる
```

} 1
} 2

1 HiSLIP インタフェース特有の機能の使用するために、IO Resource オブジェクトの IHislipInstr インタフェースの変数 (ihislipDev1) を使用しています。プログラムからは、ihislipDev1 を使用することで、LAN 接続 (HiSLIP プロトコル) の測定器に対して、リモート/ローカルを制御することができます。

2 IHislipInstr インタフェースの ControlREN メソッドを、GPIB_REN_GTL を指定して呼び出すことにより、LAN (HiSLIP プロトコル) 接続の測定器をローカル状態にしています。

【 LAN 接続 (HiSLIP プロトコル以外) の場合、RS232C 接続の場合 】

測定器のリモート/ローカルを制御するための統一された方法はありません。実現方法は測定器ごとに異なります。

測定器を LAN で接続する場合、VISA COM ライブラリでは HiSLIP 以外に、SICL-LAN、VXI-11、ソケット、のプロトコルを使用可能です。ただし、測定器によって使用できるプロトコルは異なりますので、接続できるプロトコルを確認するには、測定器のマニュアルをご参照ください。

4. Excel / VBA のテクニック

1) プログラムの実行に待ち時間を入れる

プログラムの実行に一定の待ち時間を入れたい場合、Excel / VBA から Windows の KERNEL32.dll に含まれる Sleep を呼び出す方法があります。Sleep を呼び出すには、クリック・イベントハンドラの外部に Sleep の宣言を追加します。Sleep を呼び出すと Excel 自体の実行が停止しますので、長時間を指定して Sleep を呼び出すと、ユーザからは Excel がフリーズしたように見えます。それを避けるためには、短時間の Sleep と DoEvents を繰り返す方法があります。以下は、プログラムの実行に 1 秒間の待ちを入れる例です。

```
' KERNEL32.dll 内の Sleep を呼び出すための宣言
Public Declare Sub Sleep Lib "KERNEL32.dll" (ByVal dwMilliseconds As Long)
Sub ボタン1_Click()
    Dim RM As New VisaComLib.ResourceManager
    Dim DMM As New VisaComLib.FormattedIO488
    Set DMM.IO = RM.Open("GPIB0::22::INSTR")

    For i = 1 To 10
        ' 1秒間の待ち時間
        For j = 1 To 10 ' 10回繰り返して
            Sleep (100) ' 100m秒の待ちと
            DoEvents ' DoEvents を
        Next j          ' 実施する
        DMM.WriteString "READ?" ' 測定実施
        Cells(i, 1) = DMM.ReadNumber(ASCIIType_R8) ' 測定値の読み込み
    Next i
    ' クローズ処理
    . . . .
End Sub
```

1 1 秒の待ち時間を実現するために、100 m 秒の Sleep と DoEvents を、10 回繰り返しています。

2) 測定器の応答文字列の最後から <LF> を取り除く

多くの測定器は、応答の最後に <LF> (ラインフィードキャラクタ)を終端文字として出力します。Excel / VBA で測定器から応答を読み取る場合、FormattedIO488 オブジェクトの ReadString メソッドを使用すると、終端文字の <LF> を含んだ文字列が返されます。以下は、ReadString メソッドが返す文字列から、終端文字の <LF> を取り除く例です。

```
Dim strResp As String
DMM.WriteString "*IDN?"
strResp = DMM.ReadString ' strResp の最後の文字は <LF>
strResp = Replace(strResp, vbLf, "") ' 文字列の strResp に含まれる <LF> を、Null に置き換える(取り除く)
```

3) プログラムの経過時間を調べる

Excel / VBA から Windows の KERNEL32.dll に含まれる GetTickCount を呼び出すと、ミリ秒単位で現在の時刻が取得できます。これにより、例えば、測定器の測定時間などが調査できますので、より詳細に測定器の状態を確認できます。

```
' KERNEL32.dll 内の GetTickCount を呼び出すための宣言
Public Declare Function GetTickCount Lib "KERNEL32.dll" () As Long
Sub ボタン1_Click()
    Dim RM As New VisaComLib.ResourceManager
    Dim DMM As New VisaComLib.FormattedIO488
    Set DMM.IO = RM.Open("GPIB0::22::INSTR")
    ...
    Dim startTime As Long, endTime As Long
    Dim rData() As Double

    startTime = GetTickCount ' 測定前の時刻を取得
    DMM.WriteString "READ?" ' 測定を開始し、終了すると測定値を転送する
    rData = DMM.ReadList(ASCIIType_R8)
    endTime = GetTickCount ' 測定完了後の時刻を取得

    Cells(1, 3) = endTime - startTime ' 測定にかかった時間を計算
    ...
End Sub
```

KERNEL32.dll に含まれる GetTickCount を使用するため、イベントハンドラの外部に GetTickCount の宣言を追加します。
注意) GetTickCount は、Windows が起動してから約 49 日でゼロに戻ります。約 49 日以上 Windows を再起動せずに使用する
場合、ゼロにもどった場合への対策が必要です。

4) 整数型変数のビットを判断する

測定器にステータスバイトなどのレジスタの内容を問い合わせると、測定器からは、一つの整数値が返されます。この整数値から求める情報を得るためには、整数値を 8 ビットや 16 ビットの整数として、目的のビットが 1 か 0 か、により、ステータスを判断します。

Excel / VBA でビットの判断を行う場合、3.2) 測定器のステータスを調べる (P.12) の方法に加えて、ビット演算を行う方法があります。

```
Dim resp As Integer
DMM.WriteString "*STB?" ' ステータスバイトの要求
resp = DMM.ReadNumber(ASCIIType_I2) ' ステータスバイトの内容を、2 バイト整数に読み込む
' Bit2 を確認するには、&H4(00000100) との AND をとり、結果が &H4 かどうかを確認する。
If (resp And &H4) = &H4 Then
    Dim strErr As String
    DMM.WriteString "SYST:ERR?"
    strErr = DMM.ReadString
    MsgBox "Error " & vbLf & strErr
Else
    MsgBox "No error"
End If
```

Excel / VBA では 16 進数を表記するときに &H を使用します。例えば、(16 進数 04) (2 進数 00000100) を、Excel / VBA で 16 進数であらわす場合、&H4 と記述します。

5) Excel / VBA のエラー処理

Excel / VBA から VISA COM ライブラリを呼び出したときに、VISA COM ライブラリは、実行中にエラーを検出する場合があります。例えば、

- 存在しない測定器を指定して Open した場合
- 測定器が決められた時間内に応答を返さない場合
- フォーマット変換が行えなかった場合

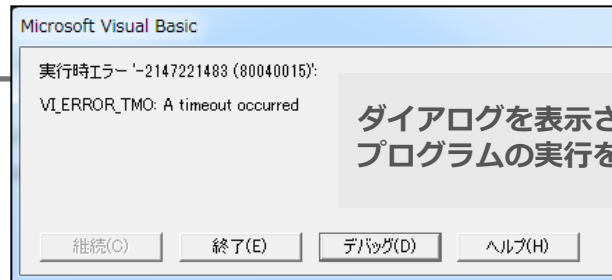
などです。

Excel / VBA が VISA COM ライブラリの実行中にエラーを検出すると、デフォルトの動作では、エラーを示すダイアログを表示させ、プログラムの実行を停止させます。

例1) デフォルトの動作

```
Sub ボタン1_Click()
    Dim RM As New VisaComLib.ResourceManager
    Dim DMM As New VisaComLib.FormattedIO488
    Set DMM.IO = RM.Open("GPIB0::22::INSTR")
    DMM.WriteString "*IDN" ' コマンドの間違い
    Dim strIDN As String
    strIDN = DMM.ReadString ' 測定器は、応答を返さないでエラーが発生
    ...
End Sub
```

エラー発生



Excel / VBA では、On Error GoTo を使用することにより、VISA COM ライブラリでエラーが発生した場合でも、プログラムの実行を継続することができます。

例2) On Error GoTo 使用例

```
Sub ボタン1_Click()
    Dim RM As New VisaComLib.ResourceManager
    Dim DMM As New VisaComLib.FormattedIO488
    Set DMM.IO = RM.Open("GPIB0::22::INSTR")

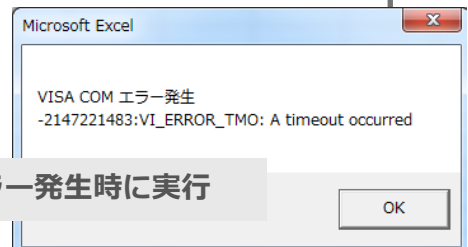
    On Error GoTo VisaComErr ' エラーが発生したら ラベル VisaComErr を実行
    ...
    DMM.WriteString "*IDN" ' コマンドの間違い
    Dim strIDN As String
    strIDN = DMM.ReadString ' 測定器は、応答を返さないでエラーが発生
    ...

    Exit Sub ' Exit Sub がないとエラーが発生しない時に VisaComErr が実行
VisaComErr:
    ' Err.Number でエラー番号、Err.Description でエラーメッセージ
    MsgBox "VISA COM エラー発生" & vbCrLf & Err.Number & "・" & Err.Description
End Sub
```

エラー発生

エラー発生時に実行するラベルを指定する

エラー発生時に実行



5. サンプル・プログラム

1) 測定器のメモリ上の測定値を Excel シートのセルに転送する

測定器を手動で操作して測定を行い、測定結果だけを Excel / VBA に転送したい場合があります。

測定値の転送だけであればプログラム作成は比較的簡単に行えますし、多くの場合、よく似たプログラムの流用が可能です。

ここでは測定器に FETCH? コマンドを送り、測定器のメモリ上のデータを読み取るサンプル・プログラムを紹介します。

例えば、34465A デジタル・マルチメータのデータ・ロギング機能を使用すると、フロントパネル操作で、1 m 秒ごとに 1000 回繰り返して DC 電圧測定を行い、測定値をメモリに保存する、という測定ができます。ロギングが完了した 34465A に FETCH? コマンドを送ると、34465A は、応答として 1000 個の測定値をカンマ区切りで返します。

以下のサンプルは、FETCH? コマンドの部分を変更することにより、他の測定器に応用が可能です。

```
Sub ボタン1_Click()
```

```
Dim RM As New VisaComLib.ResourceManager ' ResourceManager オブジェクトの作成
```

```
Dim DMM As New VisaComLib.FormattedIO488 ' FormattedIO488 オブジェクトの作成
```

```
Set DMM.IO = RM.Open("GPIB0::22::INSTR") ' IO Resource オブジェクトの作成
```

```
DMM.IO.Timeout = 10000 ' データ転送に時間がかかる場合にタイムアウトを調整
```

```
DMM.WriteString "FETCH?" ' FETCH? コマンドの出力
```

```
Dim rData() As Double
```

```
rData = DMM.ReadList(ASCIIType_R8) ' 応答の受け取り
```

```
For i = 0 To UBound(rData) ' 読み取ったデータを、A 列に格納する
```

```
Cells(i + 1, 1) = rData(i)
```

```
Next i
```

```
DMM.IO.Close ' IO Resource オブジェクトの解放
```

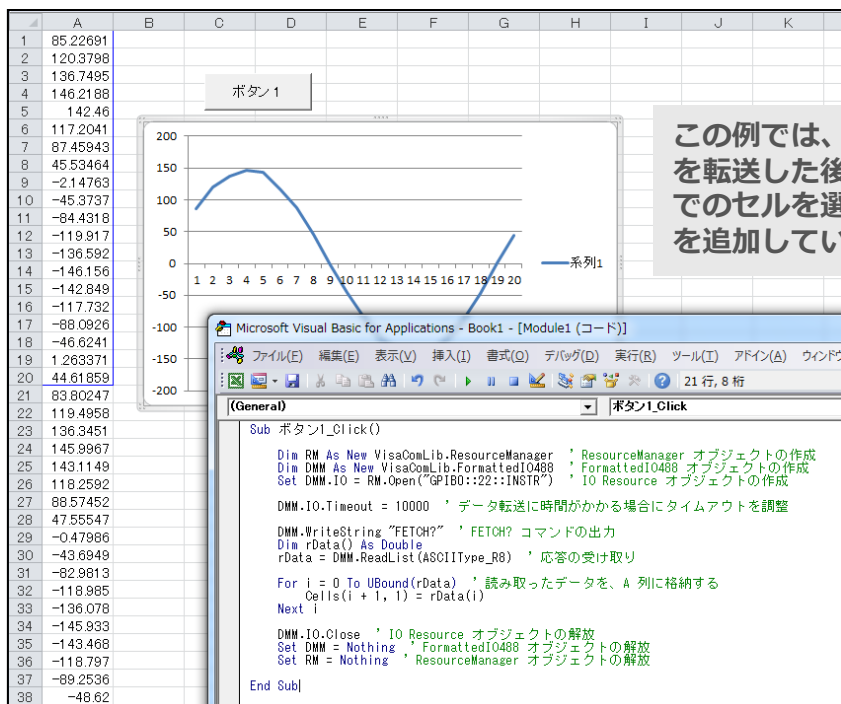
```
Set DMM = Nothing ' FormattedIO488 オブジェクトの解放
```

```
Set RM = Nothing ' ResourceManager オブジェクトの解放
```

```
End Sub
```

UBound() は、配列で指定できる添え字の最大値を返します。

ここでは、ReadList で読み取った応答をすべてセルに保存するために使用しています。



2) 測定を行い測定値を Excel シートのセルに転送する

前頁のプログラムでは、手動操作で測定を行い、プログラムでは測定値の転送だけを行っていました。

以下のサンプル・プログラムは、測定器の設定から測定開始、データ転送までプログラムで制御する例です。

34465A に DC 電圧測定、1000 Vレンジ、分解能最大、1 m 秒ごとに 1000 回測定する設定を行い、その後、測定を行い、最後に測定値を転送しています。

```
Sub ボタン1_Click()
```

```
Dim RM As New VisaComLib.ResourceManager ' ResourceManager オブジェクトの作成
```

```
Dim DMM As New VisaComLib.FormattedIO488 ' FormattedIO488 オブジェクトの作成
```

```
Set DMM.IO = RM.Open("GPIB0::22::INSTR") ' IO Resource オブジェクトの作成
```

```
DMM.WriteString "*RST" ' リセット
```

```
DMM.WriteString "*CLS" ' クリアステータス
```

```
DMM.WriteString "CONF:VOLT:DC 1000, MAX" ' DCV 1000Vレンジ、分解能最大
```

```
DMM.WriteString "SAMP:COUN 1000" ' 測定は1000回
```

```
DMM.WriteString "SAMP:SOUR TIM" ' タイマにより測定
```

```
DMM.WriteString "SAMP:TIM 1E-3" ' タイマは 1m 秒ごと
```

測定の設定

```
DMM.WriteString "SYST:ERR?" ' エラー情報の有無の問い合わせ
```

```
Dim strErr As String
```

```
strErr = DMM.ReadString ' エラー情報の取得
```

```
Cells(1, 2) = Replace(strErr, vbLf, "") ' 最後の<LF>を取り除き、B1 に保存
```

```
DMM.WriteString "INIT" ' 測定開始
```

```
DMM.WriteString "*OPC?" ' 測定が完了すると "1" を返す
```

```
Dim resp As String
```

```
resp = DMM.ReadString ' "1" を読み込む (測定完了の検出)
```

測定完了の確認

```
DMM.WriteString "FETCH?" ' FETCH? コマンドの出力
```

```
Dim rData() As Double
```

```
rData = DMM.ReadList(ASCIIType_R8) ' 応答の受け取り
```

```
For i = 0 To UBound(rData) ' 読み取ったデータを、A 列に格納する
```

```
Cells(i + 1, 1) = rData(i)
```

```
Next i
```

```
DMM.IO.Close ' IO Resource オブジェクトの解放
```

```
Set DMM = Nothing ' FormattedIO488 オブジェクトの解放
```

```
Set RM = Nothing ' ResourceManager オブジェクトの解放
```

```
End Sub
```

3) 複数の測定器を制御する

一つのプログラムで複数の測定器を制御することにより、例えば、測定器-1 で被測定物に印加する電源電圧を変更しながら測定器-2 で被測定物から出力される電圧を測定する、といった制御が可能です。

測定器-1 としては、DC 電源やファンクション・ジェネレータ、測定器-2 の例としては、デジタル・マルチメータ、周波数カウンタ、パワーメータなどが利用できます。

ここでは、キーサイト DC 電源 E36xxA の出力電圧を変更し、指定した待ち時間が経過した後で、34465A で電圧測定を行うサンプル・プログラムをご紹介します。

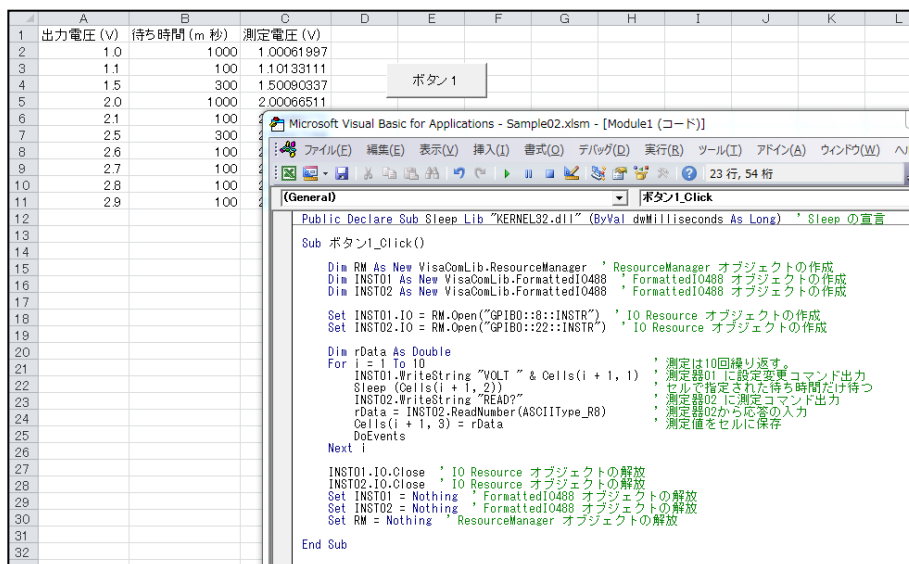
このプログラムでは、E36xxA には VOLT コマンドだけ、34465A には、READ? コマンドだけしか出力しませんので、測定に必要なその他の設定は、E36xxA、34465A とともに、あらかじめフロントパネルから手動操作で実施しておきます。

必要な設定は、E36xxA は、電流値の設定と Output On、34465A は、測定ファンクション、レンジなどの設定です。

```
Public Declare Sub Sleep Lib "KERNEL32.dll" (ByVal dwMilliseconds As Long) ' Sleep の宣言
Sub ボタン1_Click()
    Dim RM As New VisaComLib.ResourceManager ' ResourceManager オブジェクトの作成
    Dim INST01 As New VisaComLib.FormattedIO488 ' FormattedIO488 オブジェクトの作成
    Dim INST02 As New VisaComLib.FormattedIO488 ' FormattedIO488 オブジェクトの作成
    Set INST01.IO = RM.Open("GPIB0::8::INSTR") ' IO Resource オブジェクトの作成
    Set INST02.IO = RM.Open("GPIB0::22::INSTR") ' IO Resource オブジェクトの作成

    Dim rData As Double
    For i = 1 To 10 ' 測定は10回繰り返す。
        INST01.WriteString "VOLT " & Cells(i + 1, 1) ' 測定器01 に設定変更コマンド出力
        Sleep (Cells(i + 1, 2)) ' セルで指定された待ち時間だけ待つ
        INST02.WriteString "READ?" ' 測定器02 に測定コマンド出力
        rData = INST02.ReadNumber(ASCIIType_R8) ' 測定器02から応答の入力
        Cells(i + 1, 3) = rData ' 測定値をセルに保存
        DoEvents
    Next i

    INST01.IO.Close ' IO Resource オブジェクトの解放
    INST02.IO.Close ' IO Resource オブジェクトの解放
    Set INST01 = Nothing ' FormattedIO488 オブジェクトの解放
    Set INST02 = Nothing ' FormattedIO488 オブジェクトの解放
    Set RM = Nothing ' ResourceManager オブジェクトの解放
End Sub
```



6. VISA COM ライブラリ 簡易リファレンス

Open メソッド (ResourceManager オブジェクトの IResourceManager インタフェース)
VISA アドレスから IO Resource オブジェクトを作成します。

value = Open (ResourceName, mode, OpenTimeout, OptionString)

- 戻り値

value As IVisaSession

新たに作成された IO Resource オブジェクトの IVisaSession インタフェース。

- 引数

ResourceName As String

作成する IO Resource オブジェクトを指定する VISA アドレス、または、VISA Alias。

mode As VisaComLib.AccessMode (省略可能)

オープン実行時に、測定器をロックする、しないを指定。デフォルトはロックしない。

OpenTimeout As Long (省略可能)

ロックする場合の待ち時間。

OptionString As String (省略可能)

オプション指定の文字列。オープンと同時に IVisaSession の Attribute の変更が可能。

- 使い方

```
Dim RM As New VisaComLib.ResourceManager
Dim DMM As New VisaComLib.FormattedIO488
Set DMM.IO = RM.Open("GPIB0::22::INSTR")
```

IO Resource オブジェクトを作成し、作成した IO Resource オブジェクトの IVisaSession を返します。

Open メソッドは、オプションの引数を追加することで、オープンと同時に測定器をロックすることができます。測定器をロックすると、ロックした IO Resource オブジェクト以外からの制御を禁止することができます。測定器をロックする場合、OpenTimeout でロックを待つ時間を指定します。

オプションの OptionString を指定すると、Open と同時に IO Resource オブジェクトの Attribute を指定できます。

Close メソッド (IO Resource オブジェクトの IMessage インタフェース)
IO Resource オブジェクトを解放します。

Close ()

- 戻り値

なし

- 引数

なし

- 使い方

```
Dim RM As New VisaComLib.ResourceManager
Dim DMM As New VisaComLib.FormattedIO488
Set DMM.IO = RM.Open("GPIB0::22::INSTR")
. . . .
DMM.IO.Close
Set DMM = Nothing
Set RM = Nothing
```

必要に応じて、New で作成した FormattedIO488 オブジェクトと ResourceManager オブジェクトも解放します。

Timeout プロパティ (IO Resource オブジェクトの IMessage インタフェース)
データ転送のタイムアウト時間を、ミリ秒単位で参照、指定します。

```
value = Object.Timeout
Object.Timeout = value
```

- プロパティの型

value As Long
タイムアウト時間の参照、設定を、ミリ秒の単位で行います。

- 使い方

```
Dim RM As New VisaComLib.ResourceManager
Dim DMM As New VisaComLib.FormattedIO488
Set DMM.IO = RM.Open("GPIB0::22::INSTR")
DMM.IO.Timeout = 5000
```

この例は、タイムアウトを 5 秒に設定しています。

TerminationCharacterEnabled プロパティ (IO Resource オブジェクトの IMessage インタフェース)
データ入力時、終端文字の入力で読み取りを完了する、しないを参照、指定します。

```
value = Object.TerminationCharacterEnabled
Object.TerminationCharacterEnabled = value
```

- プロパティの型

value As Boolean
終端文字を読み込むことで、読み込みを完了する、しないを参照、指定します。

- 使い方

TerminationCharacter プロパティの使い方を参照

TerminationCharacter プロパティ (IO Resource オブジェクトの IMessage インタフェース)
データ入力時の終端文字を参照、指定します。

```
value = Object.TerminationCharacter
Object.TerminationCharacter = value
```

- プロパティの型

value As Byte
終端文字として使用するキャラクタを ASCII コードで指定します。

- 使い方

```
Dim RM As New VisaComLib.ResourceManager
Dim DMM As New VisaComLib.FormattedIO488
Set DMM.IO = RM.Open("GPIB0::22::INSTR")
DMM.IO.TerminationCharacter = 13 ' 終端文字を デフォルトの ASCII 10 (<LF>) から ASCII 13 (<CR>) に変更
DMM.IO.TerminationCharacterEnabled = True ' 終端文字で読み取り終了
```

入力データに ASCII の 13 (<CR>) を検出すると読み取りを完了させる設定です。デフォルトは ASCII の 10 (<LF>) です。

ReadString メソッド (FormattedIO488 オブジェクトの IFormattedIO488 インタフェース)
測定器からの応答を文字列として受け取ります。

```
value = ReadString ( )
```

- 戻り値

value As String

測定値からの応答が返ります。

- 引数

なし

- 使い方

```
Dim RM As New VisaComLib.ResourceManager
Dim DMM As New VisaComLib.FormattedIO488
Set DMM.IO = RM.Open("GPIB0::22::INSTR")
DMM.WriteString "*IDN?"
Dim strIDN As String
strIDN = DMM.ReadString
```

測定器から応答を読み込み、戻り値として String 型のデータを返します。多くの測定器は応答の最後に <LF> ラインフィードキャラクタを返します。ReadString では測定器から返された <LF> を含めて String 型のデータとして返します。

WriteString メソッド (FormattedIO488 オブジェクトの IFormattedIO488 インタフェース)
測定器に引数で指定された文字列を出力します。

```
WriteString ( data, flushAndEND )
```

- 戻り値

なし

- 引数

data As String

測定器に出力する文字列を指定します。

flushAndEND As Boolean (省略可能)

True が指定されると、出力バッファの内容を測定器に出力し、最後に終端文字、終端記号を出力します。

False が指定されると、data の内容は出力バッファにコピーされますが測定器には出力されません。

- 使い方

```
Dim data As String
data = "10, MAX"
DMM.WriteString "*RST"
DMM.WriteString "CONF:VOLT:DC ", False
DMM.WriteString data
```

第 2 引数が省略された場合、または、第 2 引数に True が指定された場合、第 1 引数で指定された文字列の最後に <LF> (ラインフィードキャラクタ、ニューラインキャラクタ、ASCII の10) を追加し、測定器に出力します。

第 2 引数に False が指定された場合、第 1 引数で指定された文字列は、FormattedIO488 の出力バッファに送られますが、測定器には出力されません。

この例では、最初の WriteString 実行時に、データは直ちに測定器に出力されますが、2 番目の WriteString の実行時には測定器には出力されません。3 番目の WriteString 実行時に、2 番目のデータと 3 番目のデータが、一度に出力されます。

ReadNumber メソッド

(FormattedIO488 オブジェクトの IFormattedIO488 インタフェース)

測定器からの応答を数値として受け取ります。

value = ReadNumber (type, flushToEnd)

- 戻り値

value As Double

測定器から読み込んだ数値データを Double 型として返します。

- 引数

type As IEEEASCIIType (省略可能)

読み取りフォーマットを指定します。

flushToEnd As Boolean (省略可能)

True の場合、入力バッファから数値を読み込み、バッファの残りをクリアします。

False の場合、入力バッファから数値を読み込み、バッファをクリアしません。

- 使い方

DMM.WriteString "READ?"

Dim dblData As Double

dblData = DMM.ReadNumber(ASCIIType_R8)

IEEEASCIIType の定義

ASCIIType_I2 '16 ビット整数

ASCIIType_I4 '32 ビット整数

ASCIIType_R4 '32 ビット実数

ASCIIType_R8 '64 ビット実数

ASCIIType_BSTR '文字列

ASCIIType_Any 'Variant 型

ASCIIType_UI1 '8 ビット符号なし整数

測定器からの応答を読み込み、数値として解釈し、Double 型のデータを返します。第 1 引数の type は、IEEEASCIIType から選択します。ReadNumber の戻り値は、type の指定に関係なく Double です。第 2 引数の flushToEnd が省略される、または、True が指定されると、読み込んだデータを先頭から数値として解釈し、残りのデータをクリアします。第 2 引数が False の場合、読み込んだデータを先頭から解釈し、残りのデータはバッファに残します。

WriteNumber メソッド

(FormattedIO488 オブジェクトの IFormattedIO488 インタフェース)

測定器に数値データを出力します。

WriteNumber (data, type, flushAndEND)

- 戻り値

なし

- 引数

data As Object (Double, Single, Long, Integer, Byte など。)

type As IEEEASCIIType (省略可能)

flushAndEnd As Boolean (省略可能)

- 使い方

Dim dblRange As Double

dblRange = 10

DMM.WriteString "VOLT:DC:RANG ", False

DMM.WriteNumber dblRange, ASCIIType_R8

数値データを指定されたフォーマットで変換し、測定器に出力します。第 1 引数には、Double、Single、Long、Integer などの数値型の変数を指定します。第 2 引数の type は、IEEEASCIIType から選択します。第 3 引数の flushAndEnd が省略される、または、True が指定されると、出力データは直ちに出力されます。False が指定されると、出力データはバッファにコピーされますが、測定器には出力されません。この例では、最初の WriteString 実行時にはデータは測定器に出力されません。2 番目の WriteNumber 実行時に、データは一度に測定器に出力されます。

ReadList メソッド

(FormattedIO488 オブジェクトの IFormattedIO488 インタフェース)

測定器から応答を読み込み、listSeparator で指定した文字で区切り、type で指定した型の配列に返します。

value = ReadList (type, listSeparator)

- 戻り値

value As Object (Double(), Single(), Long(), Integer(), Byte(), String() など、配列の変数。)

測定器から読み込んだデータの配列を返します。

- 引数

type As IEEEASCIIType (省略可能)

読み取りフォーマットを指定します。戻り値の型が決まります。

listSeparator As String (省略可能)

区切り文字の指定。デフォルトは “,” と “;” (カンマとセミコロン)。

- 使い方

```
DMM.WriteString "READ?"
Dim rData() As Double
rData = DMM.ReadList(ASCIIType_R8)
```

IEEEASCIIType の定義

ASCIIType_I2	16 ビット整数
ASCIIType_I4	32 ビット整数
ASCIIType_R4	32 ビット実数
ASCIIType_R8	64 ビット実数
ASCIIType_BSTR	文字列
ASCIIType_Any	Variant 型
ASCIIType_UI1	8 ビット符号なし整数

この例では、測定器から返されるカンマ区切りの複数の数値を、Double 型の配列に読み込んでいます。

WriteList メソッド

(FormattedIO488 オブジェクトの IFormattedIO488 インタフェース)

data の配列の内容を、type で指定されたフォーマットで変換し、listSeparator で区切り、測定器に出力します。

WriteList (data, type, listSeparator, flushAndEND)

- 戻り値

なし

- 引数

data As Double(), Single(), Long(), Integer(), Byte(), String() など、配列の変数。

測定器に出力するデータの配列。

type As IEEEASCIIType (省略可能)

出力フォーマットの指定。

listSeparator As String (省略可能)

区切り文字の指定。デフォルトは “,” (カンマ)。

flushAndEnd (省略可能)

True が指定されると、出力バッファの内容を測定器に出力し、最後に終端文字、終端記号を出力します。

False が指定されると、data の内容は出力バッファにコピーされますが測定器には出力されません。

- 使い方

```
DMM.WriteString "CONF:VOLT:DC ", False
Dim dblParam(1) As Double
dblParam(0) = 1
dblParam(1) = 0.000001
DMM.WriteList dblParam, ASCIIType_R8
```

この例では、最初に DMM.WriteString の第 2 引数を False で実行していますので、出力バッファに CONF:VOLT:DC という文字列がコピーされるだけで、この時点では、測定器にコマンドは出力されません。

次に、DMM.WriteList が実行されますが、ここで、配列 dblParam の内容、1 と 0.000001 が、すでにバッファにある CONF:VOLT:DC に追加されて、測定器に CONF:VOLT:DC 1, 0.000001<LF> が出力されます。

7. 参考資料

- IO Libraries Suite の簡易取扱説明書

[IO Libraries Suite 簡易取扱説明書 最新版]

http://www.keysight.co.jp/find/iolib_jp_quick

2015 年 6 月時点ではバージョン 17 の簡易取扱説明書にリンクしています。

[IO Libraries Suite 16.2/16.3 簡易取扱説明書] (5991-0331JAJP)

<http://literature.cdn.keysight.com/litweb/pdf/5991-0331JAJP.pdf>

- プログラミング関連資料

[Excel 2010 / VBA による測定器制御プログラム入門 VISA COM ライブラリ編] (5991-2363JAJP)

<http://literature.cdn.keysight.com/litweb/pdf/5991-2363JAJP.pdf>

[初心者向け Excel / VBA による測定器制御プログラム 超入門編] (5992-0691JAJP)

<http://literature.cdn.keysight.com/litweb/pdf/5992-0691JAJP.pdf>

[自動計測プログラム WEB]

http://www.keysight.co.jp/find/sample_program

測定器のサンプル・プログラムなどを提供しております

[これまでにリリースされたシステム開発ガイド一覧]

<http://www.keysight.com/main/editorial.jsp?cc=JP&lc=jpn&ckey=687300-3-jpn&id=687300-3-jpn>

[Test-System Development Guide] (5989-5367EN)

<http://literature.cdn.keysight.com/litweb/pdf/5989-5367EN.pdf>

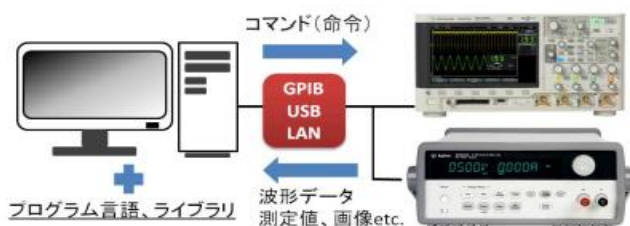
[Using VISA COM I/O API in .NET] (5989-6338EN)

<http://literature.cdn.keysight.com/litweb/pdf/5989-6338EN.pdf>

- 有償セミナーのご案内

[自動制御トレーニング・コース]

<http://www.keysight.co.jp/find/programing103>



キーサイト・テクノロジー合同会社

本社 〒192-8550 東京都八王子市高倉町9-1

計測お客様窓口

受付時間9:00-18:00(土・日・祭日を除く)

TEL ☎ 0120-421-345 (042-656-7832)

FAX ☎ 0120-421-678 (042-656-7840)

Email contact_japan@keysight.com

ホームページ www.keysight.co.jp

記載事項は変更になる場合があります。
ご発注の際はご確認ください。