VISA COM プロバイダ

菊水電子工業製直流電源 PAS シリーズ用

Version 1.0.0

ユーザーズ ガイド

October 2, 2019

備考:

【改版履歴】

バージョン	日付	内容
1.0.0	2019-05-22	初版
	2019-07-22	動作確認機器追記. 「表 1-3 CaoController クラス システム変数一覧」の説明からチャンネルに関する記述を削除(PAS シリーズにはチャンネルの機能がないため) 「2.5.制限事項」追加. 「5.付録」追加.
	2019-10-02	ドキュメント内リンクの不具合を修正.

【動作確認機器】

機種	注意事項
PAS-60-12(PIA4830 を介して接続)	

目次

1. はじめに

本書は，菊水電子工業製直流電源(PAS シリーズ)をパワーサプライ・コントローラ(PIA4800 シリーズ)を通して制御するための CAO プロバイダのユーザーズガイドです．

本書で扱う CAO プロバイダ(CaoProvKIKUSUIVISACOM.dll)を VISA COM プロバイダと呼びます．第 2 章以降ではプロバイダが提供する機能と変数について説明します．

2. プロバイダの概要

2.1. 概要

VISA COM プロバイダは，菊水電子工業製のパワーサプライ・コントローラ PIA4800 シリーズを通して直流電源 PAS シリーズに接続し，機器の設定や現在値の取得を行う CAO プロバイダです．

パワーサプライ・コントローラとの通信には，計測器接続ソフトウェアの標準仕様(VISA[1])に従って実装された VISA COM API を使用します．VISA COM API は菊水電子工業が提供する KI-VISA ライブラリをインストールすることで使用可能になります．菊水電子工業の WEB サイト(http://www.kikusui.co.jp/download/)から，バージョン 5.0.9 以上をダウンロードしインストールしてください．KI-VISA ライブラリのインストール方法については「KI-VISA Library VISA COM ガイドブック」の Setup の章をご参照ください．

以下は VISA COM プロバイダとデバイスの全体構成図です．

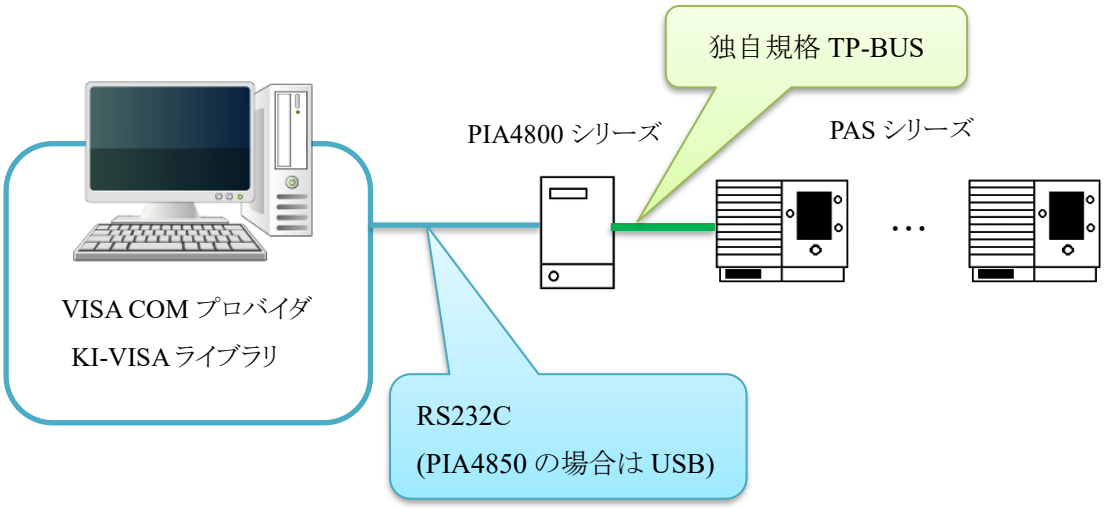

図 2-1 全体構成図

VISA COM プロバイダのファイル形式は DLL(Dynamic Link Library)であり，CAO エンジンから使用時に動的ロードされます．使用するにあたっては ORiN2SDK をインストールするか， 表 2-1 を参照して手作業でレジストリ登録を行う必要があります．

表 2-1 VISA-COM プロバイダ

ファイル名	CaoProvKIKUSUIVISACOM.dll
ProgID	CaoProv.KIKUSUI.VISACOM
レジストリ登録[2]	RegistAsm.bat CaoProvKIKUSUIVISACOM.dll
レジストリ登録の抹消	UnregistAsm.bat CaoProvKIKUSUIVISACOM.dll

[1] Virtual Instrument Software Architecture の略
[2] RegistAsm.bat および UnregistAsm.bat は{ORiN2 インストールフォルダ}¥DotNet¥BAT 以下に配置されています．

2.2. メソッド・プロパティ

2.2.1. CaoWorkspace::AddController メソッド

VISA COM プロバイダは AddController 時に接続パラメータを参照し，通信の接続を行います．

書式　AddController (<bstrCtrlName:VT_BSTR>,<bstrProvName:VT_BSTR>,
<bstrPcName:VT_BSTR>, [<bstrOption:VT_BSTR>])

<bstrCtrlName> : [in] コントローラ名
<bstrProvName> : [in] プロバイダ名．固定値 ="CaoProv.KIKUSUI.VISACOM"
<bstrPcName> : [in] プロバイダの実行マシン名（未使用）
<bstrOption> : [in] オプション文字列

以下にオプション文字列に指定するリストを示します．

表 2-2 CaoWorkspace::AddController のオプション文字列

オプション	意味
Conn=<接続パラメータ>	必須．通信形態とその接続パラメータを設定します．(参照：2.2.1.1)
Timeout[=<タイムアウト時間>]	コマンド送受信時のタイムアウト時間をミリ秒で指定します． （デフォルト：3000）

2.2.1.1. Conn オプション

以下に Conn オプションの接続パラメータ文字列を示します．ここで角括弧("[ ]")内は省略可能を示します．また，各パラメータの解説中の下線部はオプションを指定しなかったときのデフォルト値を示します．

【シリアル】

"Conn=COM:<COM Port>[:<BaudRate>]"
<COM Port> : COM ポート番号． '1' -COM1, '2' -COM2, '3' -COM3.
<BaudRate> : 通信速度． 2400, 4800, 9600, 19200.

パリティビット，データビット，ストップビット，フロー制御[3]は固定のため指定する必要はありません．

【USB】

"Conn=USB:<SerialNo>[:<ModelNo>]"
<SerialNo> : 接続対象の PIA シリーズ機器のシリアルナンバー．
<ModelNo> : 接続対象の PIA シリーズ機器の型番[4]. PIA4850.

[3] VISACOM はフロー制御(Xon/Xoff)を使用します(機器の推奨設定).
[4] 現時点では PIA4850 のみ対象としています.

使用例

```
CaoEngine caoEng;
CaoWorkspaces caoWss;
CaoWorkspace caoWs;
CaoControllers caoCtrls;
CaoController caoCtrl;

caoEng = new CaoEngine();
caoWss = caoEng.Workspaces;
caoWs = caoWss.Item(0);
caoCtrls = caoWs.Controllers;

// シリアル通信を使用して接続する場合
caoCtrl = caoWs.AddController("VISACOM_SAMPLE", "CaoProv.KIKUSUI.VISACOM", null,
                              "Conn=COM:1:19200,Timeout=5000");

// USB通信を使用して接続する場合
caoCtrl = caoWs.AddController("VISACOM_SAMPLE", "CaoProv.KIKUSUI.VISACOM", null,
                              "Conn=USB:J009999:PIA4850,Timout=5000");
```

2.2.2. CaoController::GetVariableNames プロパティ

表 2-3 に示しているシステム変数名の一覧を取得します.

2.2.3. CaoController::AddVariable メソッド

機器に対して値の設定/取得を行うための変数オブジェクトを作成します.

書式　AddVariable (<bstrVariableName:VT_BSTR>, [<bstrOption:VT_BSTR>])

<bstrVariableName>　:　[in] 変数名
<bstrOption>　:　[in] オプション文字列

使用例

```
// 出力電流取得・設定用の変数を追加
CaoVariable val;
val = caoCtrl.AddVariable("@ISET");
```

2.2.4. CaoController::Execute メソッド

CaoController クラスの Execute メソッドは, コマンドを実行するためのメソッドです. 各コマンドの詳細はコマンドリファレンスをご参照ください.

書式　Execute (<bstrCommandName:VT_BSTR>,[<vntParam:VT_VARIANT>])

< bstrCommandName>　:　[in] コマンド名
<vntParam>　:　[in] パラメータ

2.2.5. CaoVariable::get_Value プロパティ

変数名に対応するクエリメッセージを送出し，リードバックデータを文字列で返します．リードバックデータの設定[5]がヘッダおよび単位データ有りの場合，ヘッダと単位を付加したデータを返します．使用前に 2.5 制限事項を確認してください．

使用例

```
// 出力電流取得・設定用の変数を追加
CaoVariable val;
val = caoCtrl.AddVariable("@ISET");

// 現在の出力電流設定値を取得
var value = val.Value;

/*
 * ヘッダ，単位データ有りの場合のデータ
 * ISET    1.25A
 *
 * ヘッダ，単位データ無しの場合のデータ
 * 1.25
 */
```

2.2.6. CaoVariable::put_Value プロパティ

引数で渡された値をオプション指定に従い変換した後，アクセス対象のデバイスに対し書込みを行うコマンドを送出します．使用前に 2.5 制限事項を確認してください．

使用例

```
// 出力電流取得・設定用の変数を追加
CaoVariable val;
val = caoCtrl.AddVariable("@ISET");

// 出力電流値を13.20Aに設定
val.Value = "13.20";
```

[5] リードバックデータの設定は@HEAD 変数で切替可能です．

2.3. 変数一覧

2.3.1. CaoController クラス

表 2-3 CaoController クラス システム変数一覧

変数名	データ型	説明	属性	
			get	put
@MAKER_NAME	VT_BSTR	メーカー名="KIKUSUI"を返す.	○	－
@VERSION	VT_BSTR	プロバイダバージョンを返す.	○	－
@NODE	VT_BSTR	ノードアドレスを指定/取得する. ノードアドレス未設定状態の場合, "0"を返す. 設定値 : 1～36	○	○
@IOUT	VT_BSTR	接続されている直流電源装置の出力電流のモニタ値を取得する.	○	－
@ISET	VT_BSTR	接続されている直流電源装置の出力電流を設定/取得する. 設定値:接続されている電源装置の最小値～最大値	○	○
@VOUT	VT_BSTR	接続されている直流電源装置の出力電圧モニタ値を取得する.	○	－
@VSET	VT_BSTR	接続されている直流電源装置の出力電圧を設定/取得する. 設定値:接続されている電源装置の最小値～最大値	○	○
@STS	VT_BSTR	ステータスレジスタの内容を取得する.	○	－
@STB	VT_BSTR	ステータスバイトレジスタの内容を取得する.	○	－
@FAU	VT_BSTR	フォールトレジスタの内容を取得する.	○	－
@ERR	VT_BSTR	エラーレジスタの内容を取得する. 読み出し後, エラーはリセットされる.	○	－
@FUNMASK	VT_BSTR	フォールトレジスタの各ビットをセット/リセットする. 10 進数または 16 進数で指定する. 設定値 : 0～255 または 00h～FFh	○	○
@UNMASK	VT_BSTR	サービスリクエストイネーブルレジスタの各ビットをセット/リセットする. 0 進数または 16 進数で指定する. 設定値 : 0～255 または 00h～FFh	○	○

変数名	データ型	説明	属性	
			get	put
@HEAD	VT_BSTR	リードバックデータのうちヘッダ(チャンネルヘッダを含む)と単位データの有無を指定/取得する. 0 : ヘッダと単位データをつけない(デフォルト) 1 : ヘッダと単位データをつける	○	○
@LOCK	VT_BSTR	パネル操作の有効/無効を設定する. 0 : パネル操作有効(デフォルト) 1 : パネル操作無効	－	○
@OUT	VT_BSTR	接続されている直流電源装置の出力のオン/オフを設定する. 0 : 出力オフ 1 : 出力オン	－	○
@OCSET	VT_BSTR	接続されている直流電源装置の OCP 作動点を設定/取得する. 設定値 : 定格出力電流の約 10 %～約 110 %	○	○
@OVSET	VT_BSTR	接続されている直流電源装置の OVP 作動点を設定/取得する. 設定値 : 定格電圧の 10 %～110 %	○	○
@IDN	VT_ARRAY | VT_BSTR	パワーサプライ・コントローラの機種名を配列で取得する. 0 : 社名 1 : 機種名 2 : シリアル No(未使用) 3 : バージョン番号	○	－

2.4. エラーコード

VISA COM プロバイダ固有のエラーコードはありませんが, KI-VISA ライブラリで定義されたエラーコードを返す場合があります. KI-VISA のエラーについては「KI-VISA Library VISA COM ガイドブック」の VISA COM API ステータス・コードの章をご参照ください.

また, ORiN2 共通エラーについては「ORiN2 プログラミングガイド」のエラーコードの章をご参照ください.

2.5. 制限事項

直流電源 PAS シリーズに接続し機器の設定や現在値の取得を行うには, PAS シリーズの機器に対してノードアドレスを設定する必要があります. ノードアドレス未設定の場合,「@MAKER_NAME」「@ VERSION」「@NODE」以外のシステム変数の操作に失敗します. PAS シリーズの機器の電源を落すとノードアドレスが初期化されますので, 起動(再起動)時には必ずノードアドレスを再設定してください.

ノードアドレスは VISACOM プロバイダのシステム変数「@NODE」でノードアドレスを指定するか, PAS シリーズの機器のパネルを操作して設定できます.

3. コマンドリファレンス

3.1. CaoController クラス

表 3-1 CaoController クラス コマンド一覧

コマンド	機能	ページ
Clear	プログラムデータやレジスタの値を初期化する.	12
PowerOff	直流電源装置の電源をオフする.	13

3.1.1. CaoController::Execute ("Clear") コマンド

CLR メッセージを送信し, プログラムデータやレジスタの値を初期化します. また, 直流電源装置の出力をオフにします. 初期値については表 3-2 をご参照ください.

書式　Clear()

引数　：　なし

戻り値　：　なし

使用例

```
// プログラムデータ・レジスタを初期化する
caoCtrl.Execute("Clear");
```

表 3-2 プログラムデータおよびレジスタ初期値

メッセージヘッダ	初期値
NODE	0
TRM(DELIMITER)	0
FUNMASK	0h
HEAD	0
SILENT	1
*SRE(UNMASK)	0h

レジスタ	各ビット
ステータスバイトレジスタ	0
フォールトレジスタ	0
エラーレジスタ	0

3.1.2. CaoController::Execute ("PowerOff") コマンド

POW メッセージを送信し，直流電源装置の電源をオフにします．

書式　PowerOff ()

引数　：　なし

戻り値　：　なし

使用例

```
// 電源をオフにする
caoCtrl.Execute("PowerOff");
```

4. サンプルプログラム

以下に，本プロバイダを使用して PAS シリーズと通信を行い，出力電流値の読書きや Execute コマンドの実行を行うサンプルプログラム(C#)を示します．実行前に 2.5 制限事項を確認してください．

Sample **Program.cs**

```
using System;
using System.Runtime.InteropServices;
using ORiN2.interop.CAO;

namespace VISACOMSample
{
    class Program
    {
        static void Main(string[] args)
        {
            CaoEngine caoEng;
            CaoWorkspaces caoWss;
            CaoWorkspace caoWs;
            CaoControllers caoCtrls;
            CaoController caoCtrl;
            CaoVariables caoVariables;

            caoEng = new CaoEngine();
            caoWss = caoEng.Workspaces;
            caoWs = caoWss.Item(0);
            caoCtrls = caoWs.Controllers;

            // シリアル通信を使用して接続
            caoCtrl = caoWs.AddController("VISACOM_SAMPLE", "CaoProv.KIKUSUI.VISACOM", null,
                                          "Conn=COM:1:19200,Timeout=5000");

            caoVariables = caoCtrl.Variables;

            // 出力電流の変数を追加
            CaoVariable val;
            val = caoCtrl.AddVariable("@ISET");

            // 現在の出力電流設定値を取得
            var value = val.Value;

            // 出力電流値を13.20Aに設定
            val.Value = "13.20";

            // プログラムデータ・レジスタを初期化する
            caoCtrl.Execute("Clear");

            // 直流電源装置の電源をオフにする
            caoCtrl.Execute("PowerOff");

            // COMオブジェクトの解放 ----------
            // CaoVariableを解放
            caoVariables.Clear();
            Marshal.ReleaseComObject(val);
            val = null;
```

```
            // CaoVariablesを解放
            Marshal.ReleaseComObject(caoVariables);
            caoVariables = null;
            // CaoControllerを解放
            caoCtrls.Clear();
            Marshal.ReleaseComObject(caoCtrl);
            caoCtrl = null;

            // CaoControllersを解放
            caoWss.Clear();
            Marshal.ReleaseComObject(caoCtrls);
            caoCtrls = null;
            // CaoWorkspaceを解放
            Marshal.ReleaseComObject(caoWs);
            caoWs = null;
            // CaoWorkspacesを解放
            Marshal.ReleaseComObject(caoWss);
            caoWss = null;

            // CaoEngineを解放
            Marshal.ReleaseComObject(caoEng);
            caoEng = null;
        }
    }
}
```

5. 付録

5.1. デバイスメッセージ対応表

システム変数と VISA COM API を使用したデバイスメッセージの対応を以下に示します.

表 5-1 システム変数とデバイスメッセージの対応

変数名	get_Value	set_Value
@NODE	NODE?	NODE【データ】 例:NODE 5
@IOUT	IOUT?	-
@ISET	ISET?	ISET【データ】 例:ISET 12.34
@VOUT	VOUT?	-
@VSET	VSET?	VSET【データ】 例:VSET 12.34
@STS	STS?	-
@STB	*STB?	-
@FAU	FAU?	-
@ERR	ERR?	-
@FUNMASK	FUNMASK?	FUNMASK【データ】 例:FUNMASK FFh
@UNMASK	UNMASK?	UNMASK【データ】 例:UNMASK FFh
@HEAD	HEAD?	HEAD【データ】 例:HEAD 1
@LOCK	-	LOCK【データ】 例:LOCK 1
@OUT	OUT?	OUT【データ】 例:OUT 1
@OCSET	OCSET?	OCSET【データ】 例:OCSET 12.34
@OVSET	OVSET?	OVSET【データ】 例:OVSET 12.34
@IDN	*IDN?	-

5.2. 参考 URL

菊水電子工業のサイトで接続＆プログラミングガイドが公開されています. 機器マニュアルと合わせて参考にしてください.

https://www.kikusui.co.jp/kiku_manuals/P/PIA4800/index_J.html

サイトでは以下の内容を確認できます.

- 電源装置との接続マニュアル
- デバイスメッセージの解説
- PIA4800 シリーズメッセージ一覧
- レジスタについて
- KI-VISA のインストールマニュアル
- IVI-COM 計測器ドライバ・プログラミングガイド