IPPA プロバイダ

レイマック LED 照明用パルス調光コントローラー IPPA シリーズ

Version 1.0.0

ユーザーズ ガイド

September 20, 2016

【備考】

【改版履歴】

バージョン	日付	内容
1.0.0	2016-09-20	初版.
	2020-02-24	会社名変更により, ドキュメント内の会社名も変更 イマック → レイマック

【対応機器】

機種	バージョン	注意事項
IPPA-7M2		
IPPA-7M4		

目次

1. はじめに

本書は，レイマック製 LED 照明用パルス調光コントローラー(IPPA シリーズ)に対し調光制御を行う 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(CaoProvIMACIPPA.dll)を IPPA プロバイダと呼びます．

第 2 章に IPPA プロバイダの概要，実装されているメソッドを記載しています．第 3 章に調光設定コマンドおよび独自コマンドを実行するためのコマンドリファレンスを記載しています．

IPPA プロバイダで実装している通信コマンドの対応状況については，通信先となるパルス調光コントローラー(IPPA シリーズ)に依存します．通信の詳細についてはレイマックの“manual_jp_IPPA-7M4(7M2).pdf”を参照してください．

2. プロバイダの概要

2.1. 概要

IPPA プロバイダは, レイマック製パルス調光コントローラー(IPPA シリーズ)に対し TCP/IP 接続でコマンドを用いて調光制御を行う CAO プロバイダです. そのファイル形式は DLL(Dynamic Link Library)であり, CAO エンジンから使用時に動的にロードされます. IPPA プロバイダを使用するにあたっては ORiN2SDK をインストールするか, 下表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 IPPA プロバイダ

ファイル名	CaoProvIMACIPPA.dll
ProgID	CaoProv.IMAC.IPPA
レジストリ登録	regsvr32 CaoProvIMACIPPA.dll
レジストリ登録の抹消	regsvr32 /u CaoProvIMACIPPA.dll

2.2. メソッド・プロパティ

2.2.1. CaoWorkspace::AddController メソッド

IPPA プロバイダは AddController 時に通信用の接続パラメータを参照し，通信の接続を行います．(TCP クライアントとして動作します)

IPPA プロバイダではネットワーク設定の機能は実装されておりません．ネットワーク設定には，レイマックの HP で提供されている IPPA 用のサンプルソフトを使用してください．

書式　AddController(<bstrCtrlName:BSTR>,<bstrProvName:BSTR>,
<bstrPCName:BSTR>,<bstrOption:BSTR>))

bstrCtrlName : [in] コントローラ名
bstrProvName : [in] プロバイダ名．固定値 =" CaoProv.IMAC.IPPA"
bstrPCName : [in] プロバイダの実行マシン名
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	説明
Conn=<接続パラメータ>	必須．通信形態と接続パラメータ． (参照 2.2.1.1)
Timeout[=<タイムアウト時間>]	送受信時のタイムアウト時間．(ms) （デフォルト：500）

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になります．IPPA シリーズは接続ポートが 4 つ用意されており，4 ポートのうち 1 ポートを指定して接続します．

(IPPA プロバイダでは TCP/IP 接続のみサポートします)

“Conn=eth:<IP Address>[:<Port No>]”
<IP Address> : IP アドレス．
例：196.168.0.29 (192.168.0.1～192.168.255.254 の範囲で指定)
<Port No> : TCP/IP 接続ポート番号 3100, 5006 (1000～65532 で指定)

2.2.2. CaoController::Execute メソッド

IPPA のコマンドを直接送受信します. 第 1 引数にコマンド名, 第 2 引数にコマンドのパラメータを指定します. 各コマンドの詳細は 3 章コマンドリファレンスを参照してください.

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

bstrCommandName: [in] コマンド名

vntParam : [in] パラメータ

2.2.3. CaoController::AddVariable メソッド

CaoController クラスの AddVariable メソッドは, 変数オブジェクトを作成するためのメソッドです.

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

<bstrVariableName> : [in] 変数名

<bstrOption> : [in] オプション文字列(未使用)

2.2.4. CaoController::get_VariableNames プロパティ

AddVariable メソッドで指定できる変数名とシステム変数名の一覧を取得します.

2.2.5. CaoVariable::get_Value プロパティ

オブジェクトに対応している変数の値を取得します. IPPA プロバイダでは, 変数オブジェクトに対して変数の値を設定する CaoVariable:put_Value メソッドが実装されていません.

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	メーカ名=“IMAC Co., Ltd.”を返します.	○	－
@VERSION	VT_I4 | VT_ARRAY	バージョン情報. 内部の 2 つのデバイスそれぞれのファームウェアのバージョンを返します.	○	－
@ERROR_STATE	VT_I4	機器のエラー状態を返します. 0 ： 正常 1 ： 照明の発光により低下した電圧の回復途中 2 ： 回復途中に発光した. このエラーは回復途中に発光してから 0.5 秒間のみ出る 3 ： 1 と 2 の状態	○	－

2.4. エラーコード

IPPA プロバイダでは，表 2-4 の固有エラーコードが定義されています．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参照してください．

表 2-4 固有エラーコード

エラー名	エラー番号	説明
全体エラー応答	0x80100000	接続先からエラー応答を受けた場合に返されます．
受信データ欠落	0x80100001	受信データが欠落しており，解析ができなかった場合に返されます．
異なるコマンド応答	0x80100002	送信したコマンドと異なるコマンド応答を受けた場合に返されます．
複数チャンネルエラー応答	0x80100101 ～0x8010FF	複数チャンネルの設定値を同時に読み書き可能なコマンドを実行し，1 チャンネル以上のエラー応答を受けた場合に返されます（2.4.1 を参照）．

2.4.1. 複数チャンネルエラー応答

IPPA プロバイダのコマンドには，複数チャンネルの設定値を同時に読み書き可能なコマンドが実装されています（3 章を参照）．これらのコマンドでは，任意のチャンネルを任意の順番で指定した配列をパラメータとします（3.1.3 を参照）．これらを実行時に接続先から 1 チャンネル以上のエラー応答を受けた場合のエラーコードは，0x80100101 ～ 0x801001FF で表されます．このエラーコードの下 2 桁は，何番目に指定したチャンネルからエラー応答を受けたかの情報を表します．エラーを受けたチャンネルの位置をビットフラグとし，それを 16 進数に変換しています．（詳細は次ページを参照）

表 2-5 に複数チャンネルエラー応答におけるエラー発生チャンネルの位置と対応するエラーコードを示します．表の「X 番目」は，「X 番目に指定したチャンネル番号」の意味であり，「チャンネル番号 X」ではないことに注意してください．

設定値を書き込むコマンドにおいて，このエラーを受けた場合，エラー応答を受けたチャンネル以外は設定値が変更されていることに注意してください．

表 2-5 複数チャンネルエラー応答のエラーコード（4 チャンネル分）

0x801001??	エラー有無（有:1, 無:0）			
	4 番目	3 番目	2 番目	1 番目
01	0	0	0	1
02	0	0	1	0
03	0	0	1	1
04	0	1	0	0
05	0	1	0	1
06	0	1	1	0
07	0	1	1	1
08	1	0	0	0
09	1	0	0	1
0A	1	0	1	0
0B	1	0	1	1
0C	1	1	0	0
0D	1	1	0	1
0E	1	1	1	0
0F	1	1	1	1

複数チャンネルエラー応答のエラーコード E_MULTIERR の算出式

$$E\_MULTIERR = E\_MULTIERR\_MASK \mid ErrSum \quad (1)$$

$$ErrSum = (Err_1^{1} + Err_2^{2} + ... + Err_N^{N}) / 2 \quad (2)$$

E_MULTIERR_MASK : 複数チャンネルエラー応答用のマスク = 0x80100100

ErrSum : 複数チャンネル同時読み書き可能なコマンドのエラー位置ビットフラグ（16 進数表記）

Err_i : *i* 番目に指定したチャンネル番号におけるエラーの有無（エラー有:2, エラー無:0）

N : チャンネル数

例: CH 2, CH 3, CH 1 の点灯 ON/OFF を確認するコマンド CaoController.Execute(“ReadOnOffSetting”, Array(2 , 3, 1)) （3.1.3 参照）の実行時, CH 2 と CH 1 でエラー応答があった場合.

→ $ErrSum = (2^1 + 0^2 + 2^3) / 2 = 0x05$

E_MULTIERR = 0x80100100 | 0x05 = 0x80100105

3. コマンドリファレンス

本章では CaoController::Execute メソッドの各コマンドについて解説します.

表 3-1 CaoController::Execute コマンド一覧

コマンド	機能	複数チャンネル同時読出/書込	
調光設定			
ReadExecutionPattern	現在選択中の実行パターン番号を確認します.	−	P. 13
WriteExecutionPattern	実行パターンを切り替えます.	−	P. 13
ReadOnOffSetting	照明の点灯 ON/OFF の設定を読み出します.	○	P. 13
WriteOnOffSetting	照明の点灯 ON/OFF の設定を書き込みます.	○	P. 14
ReadTriggerMode	トリガーモードの設定を読み出します.	○	P. 15
WriteTriggerMode	トリガーモードの設定を書き込みます.	○	P. 15
ReadDelayTime	ディレー時間の設定を読み出します.	○	P. 16
WriteDelayTime	ディレー時間の設定を書き込みます.	○	P. 16
ReadPWMDuty	PWM の Duty 比の設定を読み出します.	○	P. 17
WritePWMDuty	PWM の Duty 比の設定を書き込みます.	○	P. 17
ReadPWMDuration	PWM 点灯継続時間の設定を読み出します.	○	P. 18
WritePWMDuration	PWM 点灯継続時間の設定を書き込みます.	○	P. 18
ReadLightingSetting	保存されている調光設定を読み出します.	○	P. 19
WriteLightingSetting	現在の調光設定を保存します.	○	P. 19
独自拡張コマンド			
ExecuteCommand	IPPA コマンドを実行します	−	P. 20
GetTimeout	タイムアウト時間の取得を行います	−	P. 20
SetTimeout	タイムアウト時間の設定を行います	−	P. 21

3.1. 調光設定

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

現在選択されている実行パターン番号を返します.

書式　ReadExecutionPattern

引数　：　なし

戻り値　：　実行パターン番号 (1～8) (VT_I4)

現在選択されている実行パターン番号を取得する例を以下に示します.

使用例

```
Dim lNum as Long
lNum = caoCtrl.Execute("ReadExecutionPattern")
' lNum : 実行パターン
```

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

実行パターンを切り替えます.

書式　WriteExecutionPattern (<lPtnNo>)

lPtnNo　：　[in] 実行パターン番号 (1～8) (VT_I4)

戻り値　：　なし

実行パターンをパターン 3 に切り替える場合の例を以下に示します.

使用例

```
caoCtrl.Execute "WriteExecutionPattern", 3
```

3.1.3. CaoController::Execute ("ReadOnOffSetting") コマンド

指定したチャンネル番号の照明の点灯 ON/OFF 設定を読み出します.

複数のチャンネルの設定を同時に読み出すことができます. 指定できるチャンネルの最大数は機器のチャンネル数に依存し, IPPA-7M2, IPPA-7M4 はそれぞれ 2, 4 個までのチャンネル番号を指定できます.

N 個のチャンネル番号 { $No_{ch,1}$, $No_{ch,2}$, ..., $No_{ch,N}$ } を指定した場合, 指定したチャンネルに対応する *N* 個の ON/OFF 設定値{ $Val_{Noch,1}$, $Val_{Noch,2}$, ..., $Val_{Noch,N}$ } がチャンネル番号の順番と同順で返されます.

書式　ReadOnOffSetting(<plCHNo>)

plCHNo　：　[in] チャンネル番号 (1～4)の配列 (VT_I4 | VT_ARRAY)
戻り値　：　<plOnOff> 点灯 ON/OFF 設定の配列(VT_I4 | VT_ARRAY)
0 : 点灯 OFF
1 : 点灯 ON

CH 2 および CH 3 の照明の点灯 ON/OFF 設定を読み出す場合の例を以下に示します.

使用例

```
Dim vntRet as Variant
vntRet = caoCtrl.Execute("ReadOnOffSetting", Array(2, 3))
‘ vntRet(0) : CH2 の ON/OFF 設定
‘ vntRet(1) : CH3 の ON/OFF 設定
```

3.1.4. CaoController::Execute ("WriteOnOffSetting") コマンド

指定したチャンネル番号の照明の ON/OFF を設定します.

複数のチャンネルの設定を同時に書き込むことができます. *N* 個のチャンネル番号 { $No_{ch,1}$, $No_{ch,2}$, ..., $No_{ch,N}$ } および *N* 個の ON/OFF 設定値{ $Val_{No_{ch,1}}$, $Val_{No_{ch,2}}$, ..., $Val_{No_{ch,N}}$ }を指定してください. また, 設定値の順番は, 対応するチャンネル番号の順番と同順にしてください.

書式　WriteOnOffSetting(<plCHNo>, <plOnOff>)

plCHNo　：　[in] チャンネル番号 (1～4)の配列 (VT_I4 | VT_ARRAY)
plOnOff　：　[in] 点灯 ON/OFF 設定の配列(VT_I4 | VT_ARRAY)
0 : 点灯 OFF
1 : 点灯 ON
戻り値　：　なし

CH 3 の照明を ON(= 1), CH 2 の照明を OFF(= 0)に設定する場合の例を以下に示します.

使用例

```
caoCtrl.Execute "WriteOnOffSetting", Array(Array(3, 2), Array(1, 0))
```

3.1.5. CaoController::Execute ("ReadTriggerMode") コマンド

指定したチャンネル番号のトリガーモード設定を読み出します. トリガーモード設定では, 内部トリガーモードと外部トリガーモードが選択できます. それぞれのモードでは以下の設定になります.

・内部トリガーモード : 80kHz で常に点灯するモード

・外部トリガーモード : 外部トリガーに同期して点灯するモード

複数のチャンネルの設定を同時に読み出すことができます(3.1.3 を参照).

書式 ReadTriggerMode(<plCHNo>)

plCHNo	:	[in] チャンネル番号 (1～4)の配列 (VT_I4 | VT_ARRAY)
戻り値	:	<plTriger> : トリガーモード設定の配列(VT_I4 | VT_ARRAY)
		0 : 内部トリガーモード
		1 : 外部トリガーモード

CH 2 および CH 3 の照明のトリガーモード設定を読み出す場合の例を以下に示します.

使用例

```
Dim vntRet as Variant
vntRet = caoCtrl.Execute("ReadTriggerMode", Array(2, 3))
' vntRet(0) : CH2 のトリガーモード
' vntRet(1) : CH3 のトリガーモード設定
```

3.1.6. CaoController::Execute ("WriteTriggerMode") コマンド

指定したチャンネル番号のトリガーモードを設定します.

複数のチャンネルの設定を同時に書き込むことができます(3.1.4 を参照).

書式 WriteTriggerMode(<plCHNo>, <plOnOff>)

plCHNo	:	[in] チャンネル番号 (1～4)の配列 (VT_I4 | VT_ARRAY)
plTriger	:	[in] トリガーモード設定の配列(VT_I4 | VT_ARRAY)
		0 : 内部トリガーモード
		1 : 外部トリガーモード
戻り値	:	なし

CH 3 を内部トリガーモード, CH 2 を外部トリガーモードに設定する場合の例を以下に示します.

使用例

```
caoCtrl.Execute "WriteTriggerMode", Array(Array(3, 2), Array(0, 1))
```

3.1.7. CaoController::Execute ("ReadDelayTime") コマンド

指定した実行パターン番号およびチャンネル番号のディレー時間を読み出します.

一つの実行パターン内の複数チャンネルの設定を同時に読み出すことができます. パターン番号 No_{ptn}, N 個のチャンネル番号 { $No_{ch,1}$, $No_{ch,2}$, ..., $No_{ch,N}$ } を指定した場合, 指定したチャンネルに対応する N 個の ON/OFF 設定値{ $Val_{No^{Ptn},\ No^{ch,1}}$, $Val_{No^{Ptn},\ No^{ch,2}}$, ..., $Val_{No^{Ptn},\ No^{ch,N}}$ } がチャンネル番号の順番と同順で返されます.

書式 ReadDelayTime(<lPtnNo>, <plCHNo>)

lPtnNo	:	[in] 実行パターン番号 (1～8) (VT_I4)
plCHNo	:	[in] チャンネル番号 (1～4)の配列 (VT_I4 | VT_ARRAY)
戻り値	:	<plDlyTime> : ディレー時間の配列(VT_I4 | VT_ARRAY)
		戻り値の範囲は 0～9999 (×10 μs)です

実行パターン 5 の CH 2 および CH 3 のディレー時間設定を読み出す場合の例を以下に示します.

使用例

```
Dim vntRet as Variant
vntRet = caoCtrl.Execute("ReadDelayTime", Array(5, Array(2, 3)))
' vntRet(0) : 実行パターン 5, CH2 のディレー時間
' vntRet(1) : 実行パターン 5, CH3 のディレー時間
```

3.1.8. CaoController::Execute ("WriteDelayTime") コマンド

指定した実行パターン番号・チャンネル番号のディレー時間を設定します.

複数のチャンネルの設定を同時に書き込むことができます. パターン番号 No_{ptn}, N 個のチャンネル番号 { $No_{ch,1}$, $No_{ch,2}$, ..., $No_{ch,N}$ } および N 個の ON/OFF 設定値{ $Val_{No^{Ptn},\ No^{ch,1}}$, $Val_{No^{Ptn},\ No^{ch,2}}$, ..., $Val_{No^{Ptn},\ No^{ch,N}}$ }を指定してください. また, 設定値の順番は, 対応するチャンネル番号の順番と同順にしてください.

書式 WriteDelayTime(<lPtnNo>, <plCHNo>, <plDlyTime>)

lPtnNo	:	[in] 実行パターン番号 (1～8) (VT_I4)
plCHNo	:	[in] チャンネル番号 (1～4)の配列 (VT_I4 | VT_ARRAY)
plDlyTime	:	[in] ディレー時間の配列(VT_I4 | VT_ARRAY)
		0～9999 (×10 μs)の範囲で指定してください
戻り値	:	なし

実行パターン 5 の CH 3 のディレー時間を 33.33 ms (3333 ×10 μs), CH 2 のディレー時間を 2.5 ms (250 ×10 μs)に設定する場合の例を以下に示します.

使用例

```
caoCtrl.Execute "WriteDelayTime", Array(5, Array(3, 2), Array(3333, 250))
```

3.1.9. CaoController::Execute ("ReadPWMDuty") コマンド

指定した実行パターン番号・チャンネル番号における PWM 制御の Duty 比を読み出します. IPPA シリーズにおいて, Duty 比は 0～100%の範囲を 256 諧調で表されます. 一つの実行パターン内の複数チャンネルの設定を同時に読み出すことができます(3.1.7 を参照).

書式　ReadPWMDuty(<lPtnNo>, <plCHNo>)

lPtnNo	:	[in] 実行パターン番号 (1～8) (VT_I4)
plCHNo	:	[in] チャンネル番号 (1～4)の配列 (VT_I4 | VT_ARRAY)
戻り値	:	<plPWMDty> : PWM の Duty 比の配列(VT_I4 | VT_ARRAY)
		戻り値の範囲は 0～255 です

パターン 5 の CH 2 および CH 3 における PWM Duty 比を読み出す場合の例を以下に示します.

使用例

```
Dim vntRet as Variant
vntRet = caoCtrl.Execute("ReadPWMDuty", Array(5, Array(2, 3)))

' vntRet(0) : 実行パターン 5, CH2 における PWM Duty 比
' vntRet(1) : 実行パターン 5, CH3 における PWM Duty 比
```

3.1.10. CaoController::Execute ("WritePWMDuty") コマンド

指定した実行パターン番号・チャンネル番号における PWM 制御の Duty 比を設定します. 複数のチャンネルの設定を同時に書き込むことができます. 一つの実行パターン内の複数チャンネルの設定を同時に読み出すことができます(3.1.8 を参照).

書式　WriteDelayTime(<lPtnNo>, <plCHNo>, <plDlyTime>)

lPtnNo	:	[in] 実行パターン番号 (1～8) (VT_I4)
plCHNo	:	[in] チャンネル番号 (1～4)の配列 (VT_I4 | VT_ARRAY)
plPWMDty	:	[in] PWM の Duty 比の配列(VT_I4 | VT_ARRAY)
		0～255 の範囲で指定してください
戻り値	:	なし

パターン 5 の CH 3 の Duty 比を 255, CH 2 の Duty 比を 20 に設定する場合の例を以下に示します.

使用例

```
caoCtrl.Execute "WriteDelayTime", Array(5, Array(3, 2), Array(255, 20))
```

3.1.11. CaoController::Execute ("ReadPWMDuration") コマンド

指定した実行パターン番号・チャンネル番号における PWM 点灯継続時間を読み出します. 一つの実行パターン内の複数チャンネルの設定を同時に読み出すことができます(3.1.7 を参照).

書式　ReadPWMDuration (<lPtnNo>, <plCHNo>)

lPtnNo	:	[in] 実行パターン番号 (1～8) (VT_I4)
plCHNo	:	[in] チャンネル番号 (1～4)の配列 (VT_I4 | VT_ARRAY)
戻り値	:	<plPWMDrtn> : PWM 点灯継続時間の配列(VT_I4 | VT_ARRAY)
		戻り値の範囲は 0～9999 (×10 μs)です

パターン 5 の CH 2 および CH 3 における PWM 点灯継続時間を読み出す場合の例を以下に示します.

使用例

```
Dim vntRet as Variant
vntRet = caoCtrl.Execute("ReadPWMDuration", Array(5, Array(2, 3)))

' vntRet(0) : 実行パターン 5, CH2 における PWM 点灯継続時間
' vntRet(1) : 実行パターン 5, CH3 における PWM 点灯継続時間
```

3.1.12. CaoController::Execute ("WritePWMDuration") コマンド

指定した実行パターン番号・チャンネル番号における PWM 制御の Duty 比を設定します. 複数のチャンネルの設定を同時に書き込むことができます. 一つの実行パターン内の複数チャンネルの設定を同時に読み出すことができます(3.1.8 を参照).

書式　WritePWMDuration (<lPtnNo>, <plCHNo>, <plDlyTime>)

lPtnNo	:	[in] 実行パターン番号 (1～8) (VT_I4)
plCHNo	:	[in] チャンネル番号 (1～4)の配列 (VT_I4 | VT_ARRAY)
plPWMDty	:	[in] PWM の Duty 比の配列(VT_I4 | VT_ARRAY)
		0～9999 (×10 μs)の範囲で指定してください
戻り値	:	なし

パターン 5 の CH 3 の PWM 点灯継続時間を 55.55 ms (5555 ×10 μs), CH 2 の PWM 点灯継続時間を 60 ms (6000 ×10 μs)に設定する場合の例を以下に示します.

使用例

```
caoCtrl.Execute "WritePWMDuration", Array(5, Array(3, 2), Array(5555, 6000))
```

3.1.13. CaoController::Execute ("ReadLightingSetting") コマンド

保存されている調光設定データのうち，指定したチャンネル番号の調光設定を今の設定値に読み出します. 読み出される設定は，以下の 5 つです.

・点灯 ON/OFF　・各パターンのディレー設定値　・トリガーモード
・各パターンの PWM Duty 比　・各パターンのPWM点灯継続時間

複数のチャンネルの設定を同時に読み出すことができます(3.1.3 を参照).

書式　ReadLightingSetting (<plCHNo>)

plCHNo　:　[in] チャンネル番号 (1～4)の配列 (VT_I4 | VT_ARRAY)
戻り値　:　なし

CH 2 および CH 3 の照明の調光設定を読み出す場合の例を以下に示します.

使用例

```
caoCtrl.Execute "ReadLightingSetting", Array(2, 3)
```

3.1.14. CaoController::Execute ("WriteLightingSetting") コマンド

指定したチャンネル番号の調光設定を保存します. 複数のチャンネルの設定を同時に書き込むことができますは(3.1.3 を参照). 保存される設定項目は 3.1.13 で読み出される項目と同じです.

書式　WriteLightingSetting (<plCHNo>)

plCHNo　:　[in] チャンネル番号 (1～4)の配列 (VT_I4 | VT_ARRAY)
戻り値　:　なし

CH 3 および CH 2 の調光設定を保存する場合の例を以下に示します.

使用例

```
caoCtrl.Execute "WriteLightingSetting", Array(Array(3, 2))
```

3.2. 独自コマンド

3.2.1. CaoController::Execute ("ExecuteCommand") コマンド

IPPA のコマンドを直接実行します. コマンド実行の成否にかかわらずコマンドの応答を取得します.

サポートする IPPA のコマンドについては, IPPA シリーズのユーザーズマニュアルを参照してください.

書式　[<vntRet> =]ExecuteCommand(< bstrCommand >)

bstrCommand　:　[in] コマンドの文字列を指定します(VT_BSTR)

vntRet　:　コマンドの応答を返します. (VT_BSTR)

IPPA のコマンドを指定し, CH 3 の照明を ON に設定する場合の例を以下に示します.

使用例

```
Dim strRet as string
strRet = caoCtrl.Execute("ExecuteCommand", "W02010001")
```

3.2.2. CaoController::Execute ("SetTimeout") コマンド

タイムアウト時間を設定します

書式　SetTimeout(<iTimeout>)

iTimeout　:　[in] タイムアウト時間　(ms) (VT_UI4)

戻り値　:　なし

使用例を示します.

使用例

```
Call caoCtrl.Execute("SetTimeout", 1000)
```

3.2.3. CaoController::Execute (″GetTimeout″) コマンド

設定されているタイムアウト時間を取得します.

書式 <uiTimeout> = GetTimeout()

引数 : [in] なし

<uiTimeout> : タイムアウト値 (ms) (VT_UI4)

例を以下に示します.

使用例

```
Dim timeout as long
timeout = caoCtrl.Execute("GetTimeout")
```