KEYENCE
LJ-V7000 プロバイダ

Version 1.0.1

ユーザーズ ガイド

June 29, 2020

備考:
プロバイダで機器と接続している間は，他のアプリケーション等で設定変更を行わないでください．

【改版履歴】

バージョン	日付	内容
1.0.0	2018-08-30	初版.
	2020-03-31	誤記修正.
1.0.1	2020-06-29	GetError コマンドの引数に 0 を入れるとアプリケーションエラーになる不具合を修正

【対応機器】

機種	バージョン	注意事項

目次

1. はじめに

本書は KEYENCE 社製 2 次元レーザ変位計 LJ-V7000 シリーズの CAO プロバイダである, LJ-V7000 プロバイダのユーザーズガイドです.

LJ-V7000 プロバイダは Ethernet 接続された LJ-V7000 シリーズコントローラと通信し, 測定の制御や測定結果の取得を行います.

本書では, この LJ-V7000 プロバイダの機能と実装されているメソッドについて説明します.

2. プロバイダの概要

2.1. 概要

LJ-V7000 プロバイダは, CaoController::Execute により, KEYENCE 社製通信ライブラリのコマンドを実行することにより, 通信機能を提供しています.

表 2-1 LJ-V7000 プロバイダ

ファイル名	CaoProvKEYENCELJV7000.dll
ProgID	CaoProv.KEYENCE.LJ-V7000
レジストリ登録	regsvr32 CaoProvKEYENCELJV7000.dll
レジストリ登録の抹消	regsvr32 /u CaoProvKEYENCELJV7000.dll

表 2-2 依存モジュール

No.	ファイル名	説明
1	LJV7_IF.dll	KEYENCE 社製 Ethernet, USB 通信用ライブラリ
2	vcredist_x86.exe	Microsoft Visual C++ 2008 再頒布可能パッケージ (x86) (プロバイダ登録前にインストールする必要があります)

2.2. メソッド・プロパティ

2.2.1. CaoWorkspace::AddController メソッド

KEYENCE のライブラリの初期化, 製品との接続処理を実行します.
最大 6 台までの機器を接続することが可能です.

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

bstrCtrlName	:	[in]	CAO コントローラ名 接続単位で重複しない任意の文字列を指定します. ※異なるアプリケーションや別 PC から同一の名前を指定した場合はエラー(0x80000205)になります. 空文字列("")を指定した場合, CAO エンジンが自動的にユニークな CAO コントローラ名を割り当てます.
bstrProvName	:	[in]	プロバイダ名. 固定値 ="CaoProv.KEYENCE.LJ-V7000".
bstrPcName	:	[in]	プロバイダの実行マシン名 同一マシン上の場合は空文字列("")を指定します.
bstrOption	:	[in]	オプション文字列＝"<オプション 1>,<オプション 2>,…"

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 2-3 CaoWorkspace::AddController のオプション文字列

オプション	説明
Conn=eth:<IP アドレス>[:ポート番号]	必須. 接続対象の変位計のコントローラの IP アドレスを指定します. ポート番号のみ省略可能です. (デフォルト:24691) 例:"Conn=eth:192.168.0.1:24691"
DeviceID=<ID>	必須. 接続対象のコントローラに固有の ID を設定します. ID は 0~5 の範囲で設定し, 複数接続する場合には重複の無いようにしてください 例:"DeviceID=1"
Head=<ヘッド数>	必須. 接続対象のコントローラに接続されているヘッドの数(1 または 2)を指定します. 例:"Head=1"
HSDComm=<高速データ通信 ON/OFF>[:<高速データ通信用ポート番号>:<1 度にまとめて送信す	AddController 時に高速データ通信を開始するかどうかを指定します. 高速データ通信 OFF(1 番目のパラメータが 0)の場合, 2 番目以降のパラメータを省略することができます. (デフォル

るプロファイル数>]	ト:0) <高速データ通信 ON/OFF>: 0: OFF 1: ON 例: "HSDComm=0" "HSDComm=1:24692:10"

使用例

```
' 使用例プログラム
Dim caoCtrl As Object
caoCtrl = caoWs.AddController("LJV7",
                              "CaoProv.KEYENCE.LJ-V7000",
                              "",
                              "conn=192.168.0.1:24691, DeviceID=1, Head=1
                                    , HSDComm=1:24692:10")
```

2.2.2. CaoController::Execute メソッド

コマンドの送受信を行います. 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. コマンドリファレンスを参照してください.

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

bstrCommandName	:	[in]	コマンド名
vntParam	:	[in]	コマンドの引数

2.3. イベント

2.3.1. CaoController::OnMessage イベント

高速データ通信を行った際に出力されたデータを受信します.

機器のサンプリング周波数が高い場合, データ受信が遅延する可能性があります. その場合は下記のいずれかの手段により, 複数プロファイルをまとめて受信するようにしてください.

1. StartHighSpeedDataCommunication コマンドの 2 つ目の引数(dwProfileCnt)の値を大きくする.
2. AddController の HSDComm オプションの 3 つ目の値(1 度にまとめて送信するプロファイル数)を大きくする.

表 2-4 イベント種別一覧

種別	説明	Value の内容
0x (x: DeviceID)	プロファイル情報	DeviceID オプションで指定した ID のコントローラで計測されたプロファイルについて，表 2-5 の順に格納された情報を受信します． 同じ DeviceID のコントローラからプロファイルデータを受信するのと同時にプロファイルデータも受信します． 高速データ通信が継続している間，値は変化しません．
1x (x: DeviceID)	プロファイルデータ	DeviceID オプションで指定した ID のコントローラで計測されたプロファイルのデータを取得します． StartHighSpeedDataCommunication コマンドの引数または HSDComm オプションで指定した数のプロファイルのデータが蓄積するとデータを受信します． プロファイルのデータ数については，「LJ-V7000 シリーズ 通信ライブラリ リファレンスマニュアル」9.2.9.6 項を参照してください．

表 2-5 プロファイル情報の格納データ (VT_ARRAY|VT_VARIANT)

Index	内容	データ型
0	1 単位のデータに格納されているプロファイル数	VT_UI1
1	プロファイル圧縮(時間軸)の ON/OFF	VT_UI1
2	1 プロファイル中のデータ点数	VT_UI2
3	1 点目の X 座標	VT_I4
4	データ点の X 方向の間隔	VT_I4

2.4. エラーコード

LJ-V7000 プロバイダでは，以下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 2-6 固有エラーコード

エラー名	エラー番号	説明
ライブラリエラー	0x8010xxxx	通信ライブラリのエラーです． エラーコードの内容については「LJ-V7000 シリーズ 通信ライブラリ」を参照してください．

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. 各コマンドの詳細動作については KEYENCE 社の「LJ-V7000 シリーズ 通信ライブラリ リファレンスマニュアル」を参照してください.

表 3-1 CaoController::Execute コマンド一覧

コマンド	LJ-V7000 コマンド	機能	ページ
システム制御			
GetError	GetError	コントローラのシステムエラー情報を取得します	11
ClearError	ClearError	コントローラのシステムエラーを解除します	11
測定制御			
Trigger	Trigger	トリガを発行します	12
StartMeasure	StartMeasure	バッチ測定開始します	12
StopMeasure	StopMeasure	バッチ測定終了します	13
AutoZero	AutoZero	ZERO 入力を行います(高機能モード専用)	13
Timing	Timing	TIMING 入力を行います(高機能モード専用)	13
Reset	Reset	RESET 入力を行います(高機能モード専用)	14
ClearMemory	ClearMemory	変位計のコントローラ内部のメモリをクリアします	14
設定変更/読み出し関連			
GetTime	GetTime	コントローラの日時を取得します	15
GetActiveProgram	GetActiveProgram	アクティブプログラムを取得します	15
ChangeActiveProgram	ChangeActiveProgram	アクティブプログラムを切り替えます	15
測定結果の取得			
GetMeasurementValue	GetMeasurementValue	OUT 測定の現在値を取得します	16
GetProfile	GetProfile	指定したプロファイルを取得します	16
GetBatchProfile	GetBatchProfile	指定したバッチ測定のプロファイルを取得します	18
GetProfileAdvance	GetProfileAdvance	指定したプロファイルを取得します	19
GetBatchProfileAdvance	GetBatchProfileAdvance	指定したバッチ測定のプロファイルを取得します	20
ストレージ機能関連			
StartStorage	StartStorage	ストレージを開始します	22
StopStorate	StopStorate	ストレージを終了します	22
GetStorageStatus	GetStorageStatus	ストレージの状態を取得します	22
GetStorageData	GetStorageData	ストレージ内の OUT 測定値を取得します	23
GetStorageProfile	GetStorageProfile	ストレージ内のプロファイデータを取得します	24
GetStorageBatchProfile	GetStorageBatchProfile	ストレージ内のバッチプロファイルデータを取得します	26
高速データ通信関連			

StartHighSpeed DataCommunication	HighSpeedDataEthernet CommunicationInitialize PreStartHighSpeedData Communication StartHighSpeedData Communication	高速データ通信を開始します	27
StopHighSpeed DataCommunication	StopHighSpeedData Communication HighSpeedData CommunicationFinalize	高速データ通信を停止します	28

3.1. システム制御

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

コントローラで発生しているシステムエラーコードを, 指定した数だけ取得します.

書式　GetError (<byRcvMax>)

byRcvMax　：　システムエラー取得する最大個数 (VT_UI1)

戻り値　：　コントローラで発生しているシステムエラーコードの数と, 取得したエラーコード (VT_ARRAY|VT_VARIANT)
データは表 3-2 のように格納されています.

表 3-2 GetError 取得データ

Index	内容	データ型
0	コントローラ内で発生しているシステムエラーの個数	VT_UI1
1~<byRcvMax>	取得したエラーコード	VT_UI2

使用例

```
' 使用例プログラム
Dim retVal As Variant
retVal = caoCtrl.Execute("GetError", 5)
```

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

指定したシステムエラーコードを解除します. 解除可能なエラーコードについては「LJ-V7000 シリーズ 通信ライブラリ リファレンスマニュアル」9.2.3 項を参照してください.

書式　ClearError (<wErrCode>)

wErrCode : 解除したいシステムエラーコード (VT_UI2)
戻り値 : 無し

使用例

```
' 使用例プログラム
caoCtrl.Execute "ClearError", &H0085
```

3.2. 測定制御

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

トリガーを発行します.

書式 Trigger( )

引数 : なし
戻り値 : なし

使用例

```
' 使用例プログラム
caoCtrl.Execute "Trigger"
```

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

バッチ測定を開始します.

書式 StartMeasure( )

引数 : なし
戻り値 : なし

使用例

```
' 使用例プログラム
caoCtrl.Execute "StartMeasure"
```

3.2.3. CaoController::Execute("StopMeasure") コマンド

バッチ測定を終了します.

書式　StopMeasure( )

引数　：なし
戻り値　：なし

使用例

```
' 使用例プログラム
caoCtrl.Execute "StopMeasure"
```

3.2.4. CaoController::Execute("AutoZero") コマンド

指定した OUT 測定に対しオートゼロ要求を発行します.

書式　AutoZero (<byOnOff>, <dwOut>)

byOnOff　：ON/OFF 指定 (VT_UI1)
dwOut　：処理対象の OUT 測定(ビット指定) (VT_UI4)
　例:OUT2, 5 が対象の場合, dwOut=0b0000000000010010(0x0012)
戻り値　：なし

使用例

```
' 使用例プログラム
caoCtrl.Execute "AutoZero", Array(1, &H0012)
```

3.2.5. CaoController::Execute("Timing") コマンド

指定した OUT 測定に対しタイミング要求を発行します.

書式　Timing (<byOnOff>, <dwOut>)

byOnOff　：ON/OFF 指定 (VT_UI1)
dwOut　：処理対象の OUT 測定(ビット指定) (VT_UI4)
　例:OUT2, 5 が対象の場合, dwOut=0b0000000000010010(0x0012)

戻り値 : なし

使用例

```
' 使用例プログラム
caoCtrl.Execute "Timing", Array(1, &H0012)
```

3.2.6. CaoController::Execute("Reset") コマンド

指定した OUT 測定に対しリセット要求を発行します.

書式 Reset (<dwOut>)

dwOut : 処理対象の OUT 測定(ビット指定) (VT_UI4)
例:OUT2, 5 が対象の場合, dwOut=0b0000000000010010(0x0012)
戻り値 : なし

使用例

```
' 使用例プログラム
caoCtrl.Execute "AutoZero", &H0012
```

3.2.7. CaoController::Execute("ClearMemory") コマンド

コントローラ内に蓄積されたデータをクリアします.

書式 ClearMemory ( )

引数 : なし
戻り値 : なし

使用例

```
' 使用例プログラム
caoCtrl.Execute "ClearMemory"
```

3.3. 設定変更/読み出し関連

3.3.1. CaoController::Execute("GetTime") コマンド

コントローラの日時を取得します.

書式　GetTime( )

引数　：　なし

戻り値　：　コントローラの日時 (VT_DATE)

使用例

```
' 使用例プログラム
Dim retVal As Variant
retVal = caoCtrl.Execute("GetTime")
```

3.3.2. CaoController::Execute("ChangeActiveProgram") コマンド

アクティブプログラム No.を切り替えます.

書式　ChangeActiveProgram(<byProgNo>)

byProgNo　：　切替後のプログラム No. (VT_UI1)

戻り値　：　なし

使用例

```
' 使用例プログラム
caoCtrl.Execute("ChangeActiveProgram", 2)
```

3.3.3. CaoController::Execute("GetActiveProgram") コマンド

現在のアクティブプログラム No.を取得します.

書式　GetActiveProgram( )

引数　：　なし

戻り値　：　アクティブプログラム No. (VT_UI1)

使用例

```
' 使用例プログラム
Dim retVal As Variant
retVal = caoCtrl.Execute("GetActiveProgram")
```

3.4. 測定結果の取得

3.4.1. CaoController::Execute("GetMeasurementValue") コマンド

最新の測定値を取得します.

書式　GetMeasurementValue ( )

引数　：なし

戻り値　：全ての OUT 測定データ (VT_ARRAY|VT_VARIANT)
データは表 3-3 のように格納されています.

表 3-3 GetMeasurementValue 取得データ

Index	内容	データ型
0	OUT1 の測定値が有効か無効か	VT_UI1
1	OUT1 の交差判定結果	VT_UI1
2	OUT1 の測定値	VT_R4
…	…	…
45	OUT16 の測定値が有効か無効か	VT_UI1
46	OUT16 の交差判定結果	VT_UI1
47	OUT16 の測定値	VT_R4

使用例

```
' 使用例プログラム
Dim retVal As Variant
Dim outVal As Single
retVal = caoCtrl.Execute("GetMeasurementValue")
outVal = retVal(47)     'Get measurement value of OUT16
```

3.4.2. CaoController::Execute("GetProfile") コマンド

指定した, 高速モードのプロファイルを取得します.

書式　GetProfile (<byTargetBank>,<byPosMode>,<dwGetProfNo>,<byGetProfCnt>,<byErase>)

byTargetBank : アクティブ面/非アクティブ面のどちらから取得するか
(VT_UI1)
byPosMode : プロファイル取得位置の指定方法
(VT_UI1)
dwGetProfNo : byPosMode で 2 を指定した場合に, 取得対象のプロファイル No.
(VT_UI4)
byGetProfCnt : 読み出すプロファイル数 (VT_UI1)
byErase : 読み出したプロファイルとそれ以前のプロファイルを消去するか
(VT_UI1)
戻り値 : プロファイルデータ (VT_ARRAY|VT_VARIANT)
データは表 3-4 のように格納されています.

表 3-4 GetProfile 取得データ

Index1	Index2	内容	データ型
0	0	1 単位のデータに格納されているプロファイル数	VT_UI1
	1	プロファイル圧縮(時間軸)の ON/OFF	VT_UI1
	2	1 プロファイル中のデータ点数	VT_UI2
	3	1 点目の X 座標	VT_I4
	4	データ点の X 方向の間隔	VT_I4
	5	取得時点の最新プロファイル No.	VT_UI4
	6	コントローラが保持する最古のプロファイルのプロファイル No.	VT_UI4
	7	読み出した中で最古のプロファイルのプロファイル No.	VT_UI4
	8	今回読み出したプロファイル数	VT_UI1
1	0~	プロファイルデータ 取得したプロファイルデータが順に格納されています. 1 単位のデータ数及びデータ格納順については「LJ-V7000 シリーズ 通信ライブラリ リファレンスマニュアル」9.2.9.6, 9.2.9.7 項を参照してください.	VT_I4

使用例

```
' 使用例プログラム
Dim retVal As Variant
Dim infoVal As Variant
```

```
Dim profVal As Variant
Dim data As Integer
retVal = caoCtrl.Execute("GetProfile", Array(0, 0, 0, 10, 0))
infoVal = retVal(0)
profVal = retVal(1)
data = profVal(infoVal(0) * infoVal(2) * 9) 'Get 1st data of 10th profile
```

3.4.3. CaoController::Execute("GetBatchProfile") コマンド

指定した, 高速モードのバッチ測定のプロファイルを取得します.

書式　GetBatchProfile (<byTargetBank>,<byPosMode>,<byGetBatchNo>,
<dwGetProfNo>,<byGetProfCnt>,<byErase>)

byTargetBank	:	アクティブ面/非アクティブ面のどちらから取得するか (VT_UI1)
byPosMode	:	バッチ位置の指定方法 (VT_UI1)
byGetBatchNo	:	byPosMode で 2 を指定した場合に, 取得対象のプロファイル No. (VT_UI4)
dwGetProfNo	:	バッチ内の取得開始プロファイル No. (VT_UI4)
byGetProfCnt	:	読み出すプロファイル数 (VT_UI1)
byErase	:	読み出したプロファイルとそれ以前のプロファイルを消去するか (VT_UI1)
戻り値	:	プロファイルデータ (VT_ARRAY|VT_VARIANT) データは表 3-5 のように格納されています.

表 3-5 GetBatchProfile 取得データ

Index1	Index2	内容	データ型
0	0	1 単位のデータに格納されているプロファイル数	VT_UI1
	1	プロファイル圧縮(時間軸)の ON/OFF	VT_UI1
	2	1 プロファイル中のデータ点数	VT_UI2
	3	1 点目の X 座標	VT_I4
	4	データ点の X 方向の間隔	VT_I4
	5	取得時点の最新バッチ No.	VT_UI4
	6	最新バッチ内のプロファイル数	VT_UI4
	7	コントローラが保持する最古のバッチ No.	VT_UI4

	8	コントローラが保持する最古のバッチ内のプロファイル数	VT_UI4
	9	今回読み出したバッチ No.	VT_UI4
	10	今回読み出したバッチ内のプロファイル数	VT_UI4
	11	読み出した中で最古のプロファイルが，バッチ内の何番目のプロファイルか	VT_UI4
	12	今回読み出したプロファイル数	VT_UI1
	13	最新バッチ No.のバッチ測定が完了しているか	VT_UI1
1	0~	プロファイルデータ 取得したプロファイルデータが順に格納されています. 1 単位のデータ数及びデータ格納順については「LJ-V7000 シリーズ 通信ライブラリ リファレンスマニュアル」9.2.9.6, 9.2.9.7 項を参照してください.	VT_I4

使用例

```
' 使用例プログラム
Dim retVal As Variant
Dim infoVal As Variant
Dim profVal As Variant
Dim data As Integer
retVal = caoCtrl.Execute("GetBatchProfile", Array(0, 0, 0, 0, 10, 0))
infoVal = retVal(0)
profVal = retVal(1)
data = profVal(infoVal(0) * infoVal(2) * 9) 'Get 1st data of 10th profile
```

3.4.4. CaoController::Execute("GetProfileAdvance") コマンド

指定した，高機能モードのプロファイルを取得します.

書式　GetProfileAdvance ( )

引数　：なし

戻り値　：プロファイルデータ (VT_ARRAY|VT_VARIANT)
データは表 3-6 のように格納されています.

表 3-6 GetProfileAdvance 取得データ

Index1	Index2	内容	データ型
0	0	1 単位のデータに格納されているプロフ	VT_UI1

		ァイル数	
	1	プロファイル圧縮(時間軸)の ON/OFF	VT_UI1
	2	1 プロファイル中のデータ点数	VT_UI2
	3	1 点目の X 座標	VT_I4
	4	データ点の X 方向の間隔	VT_I4
1	0~	プロファイルデータ 取得したプロファイルデータが順に格納されています. 1 単位のデータ数及びデータ格納順については「LJ-V7000 シリーズ 通信ライブラリ リファレンスマニュアル」9.2.9.6, 9.2.9.7 項を参照してください.	VT_I4
2	0~47	OUT 測定データ **表 3-4** と同様のデータ順で格納されています	VT_UI1 または VT_R4

使用例

```
' 使用例プログラム
Dim retVal As Variant
Dim infoVal As Variant
Dim outVal As Variant
Dim data As Integer
retVal = caoCtrl.Execute("GetProfileAdvance")
infoVal = retVal(0)
outVal = retVal(2)
data = outVal(3 * 4 + 2) 'Get measurement value of OUT5
```

3.4.5. CaoController::Execute("GetBatchProfileAdvance") コマンド

指定した, 高機能モードのバッチ測定のプロファイルを取得します.

書式　GetBatchProfileAdvance (<byPosMode>,<byGetBatchNo>,<dwGetProfNo>,<byGetProfCnt>)

byPosMode : バッチ位置の指定方法
(VT_UI1)

byGetBatchNo : byPosMode で 2 を指定した場合に, 取得対象のプロファイル No.
(VT_UI4)

dwGetProfNo : バッチ内の取得開始プロファイル No. (VT_UI4)

byGetProfCnt : 読み出すプロファイル数 (VT_UI1)

戻り値 : プロファイルデータ (VT_ARRAY|VT_VARIANT)
データは表 3-7 のように格納されています.

表 3-7 GetBatchProfileAdvance 取得データ

Index1	Index2	内容	データ型
0	0	1 単位のデータに格納されているプロファイル数	VT_UI1
	1	プロファイル圧縮(時間軸)の ON/OFF	VT_UI1
	2	1 プロファイル中のデータ点数	VT_UI2
	3	1 点目の X 座標	VT_I4
	4	データ点の X 方向の間隔	VT_I4
	5	今回読み出したバッチ No.	VT_UI4
	6	今回読み出したバッチ内のプロファイル数	VT_UI4
	7	読み出した中で最古のプロファイルが, バッチ内の何番目のプロファイルか	VT_UI4
	8	今回読み出したプロファイル数	VT_UI1
1	0~	プロファイルデータ 取得したプロファイルデータが順に格納されています. 1 単位のデータ数及びデータ格納順については「LJ-V7000 シリーズ 通信ライブラリ リファレンスマニュアル」9.2.9.6, 9.2.9.7 項を参照してください.	VT_I4
2	0~	各プロファイルに対する OUT 測定データ 下記の要素数のデータが格納されています <16OUT 分のデータ(**表 3-4** と同様のデータ順)> ×<読み出したプロファイル数>	VT_UI1 または VT_R4
3	0~47	取得対象のバッチに対する OUT 測定データ **表 3-4** と同様のデータ順で格納されています	VT_UI1 または VT_R4
4	0~47	コマンド実行時点での最新 OUT 測定データ **表 3-4** と同様のデータ順で格納されています	VT_UI1 または VT_R4

使用例

```
' 使用例プログラム
Dim retVal As Variant
Dim infoVal As Variant
Dim outVal As Variant
Dim data As Integer
retVal = caoCtrl.Execute("GetBatchProfileAdvance", Array(0, 0, 0, 10))
infoVal = retVal(0)
outVal = retVal(2)
```

```
data = outVal(48 * 9 + 3 * 4 + 2) 'Get measurement value of OUT5 of 10th profile
```

3.5. ストレージ機能関連

3.5.1. CaoController::Execute("StartStorage") コマンド

ストレージを開始します.

書式　StartStorage( )

引数　：　なし

戻り値　：　なし

使用例

```
' 使用例プログラム
caoCtrl.Execute "StartStorage"
```

3.5.2. CaoController::Execute("StopStorage") コマンド

ストレージを終了します.

書式　StopStorage( )

引数　：　なし

戻り値　：　なし

使用例

```
' 使用例プログラム
caoCtrl.Execute "StopStorage"
```

3.5.3. CaoController::Execute("GetStorageStatus") コマンド

ストレージの状態を取得します.

書式　GetStorageStatus (<dwReadArea>)

dwReadArea　：　読み出しストレージ面

(VT_UI4)

戻り値 : ストレージ情報 (VT_ARRAY|VT_VARIANT)
データは表 3-8 のように格納されています.

表 3-8 GetStorageStatus 取得データ

Index	内容	データ型
0	ストレージの空き状態	VT_UI1
1	対象ストレージ面のプログラム No.	VT_UI1
2	ストレージ対象	VT_UI1
3	ストレージ点数	VT_UI4
4	ストレージの面数	VT_UI4
5	アクティブストレージ面の No.	VT_UI4

使用例

```
' 使用例プログラム
Dim retVal As Variant
Dim statusVal As Byte
retVal = caoCtrl.Execute("GetStorage", 0)
statusVal = retVal(2)   'Get storage target
```

3.5.4. CaoController::Execute("GetStorageData") コマンド

ストレージの OUT 測定データを取得します.

書式 GetStorageData (<dwSurface >,< dwStartNo >,< dwDataCnt >)

dwSurface : 読み出しストレージ面 (VT_UI4)
dwStartNo : 読み出しを開始するデータ No. (VT_UI4)
dwDataCnt : 読み出したいデータ点数 (VT_UI4)
戻り値 : ストレージの OUT 測定データ (VT_ARRAY|VT_VARIANT)
データは表 3-9 のように格納されています.

表 3-9 GetStorageData 取得データ

Index1	Index2	内容	データ型
0	0	ストレージの空き状態	VT_UI1
	1	対象ストレージ面のプログラム No.	VT_UI1

	2	ストレージ対象	VT_UI1
	3	ストレージ点数	VT_UI4
	4	読み出し開始したデータ No.	VT_UI4
	5	今回読み出したデータ点数	VT_UI4
	6	基準時刻	VT_DATE
1	0~	各データの基準時刻から 10ms 単位でカウントされた値 読み出したデータ点数分，カウンタ値のが格納されています	VT_UI4
2	0~	ストレージの OUT 測定データ 下記の要素数のデータが格納されています <16OUT 分のデータ(**表 3-4** と同様のデータ順)> ×<読み出したデータ点数>	VT_UI1 または VT_R4

使用例

```
' 使用例プログラム
Dim retVal As Variant
Dim outVal As Variant
retVal = caoCtrl.Execute("GetStorageData", Array(0, 0, 10))
outVal = retVal(48 * 6 + 3 * 1 + 1) 'Get judge result of OUT2 of 7th profile
```

3.5.5. CaoController::Execute("GetStorageProfile") コマンド

ストレージのプロファイルデータを取得します.

書式　GetStorageProfile (<dwSurface >,< dwStartNo >,< dwDataCnt >)

dwSurface	:	読み出しストレージ面 (VT_UI4)
dwStartNo	:	読み出しを開始するデータ No. (VT_UI4)
dwDataCnt	:	読み出したいデータ点数 (VT_UI4)
戻り値	:	ストレージのプロファイルデータ (VT_ARRAY|VT_VARIANT) データは表 3-10 のように格納されています.

表 3-10 GetStorageProfile 取得データ

Index1	Index2	内容	データ型
0	0	ストレージの空き状態	VT_UI1
	1	対象ストレージ面のプログラム No.	VT_UI1
	2	ストレージ対象	VT_UI1

	3	ストレージ点数	VT_UI4
	4	読み出し開始したデータ No.	VT_UI4
	5	今回読み出したデータ点数	VT_UI4
	6	基準時刻	VT_DATE
	7	1 単位のデータに格納されているプロファイル数	VT_UI1
	8	プロファイル圧縮(時間軸)の ON/OFF	VT_UI1
	9	1 プロファイル中のデータ点数	VT_UI2
	10	1 点目の X 座標	VT_I4
	11	データ点の X 方向の間隔	VT_I4
1	0~	各データの基準時刻から 10ms 単位でカウントされた値 読み出したデータ点数分, カウンタ値のが格納されています	VT_UI4
2	0~	プロファイル取得時点の最新 OUT 測定結果 下記の要素数のデータが格納されています <16OUT 分のデータ(**表 3-4** と同様のデータ順)> ×<読み出したデータ点数>	VT_UI1 または VT_R4
3	0~	プロファイルデータ 取得したプロファイルデータが順に格納されています. 1 単位のデータ数及びデータ格納順については「LJ-V7000 シリーズ 通信ライブラリ リファレンスマニュアル」9.2.9.6, 9.2.9.7 項を参照してください.	VT_I4
4	0~	各プロファイルに対する OUT 測定結果 下記の要素数のデータが格納されています <16OUT 分のデータ(**表 3-4** と同様のデータ順)> ×<読み出したデータ点数>	VT_UI1 または VT_R4

使用例

```
' 使用例プログラム
Dim retVal As Variant
Dim infoVal As Variant
Dim profVal As Variant
Dim data As Integer
retVal = caoCtrl.Execute("GetStorageProfile", Array(0, 0, 10))
infoVal = retVal(0)
profVal = retVal(1)
data = profVal(infoVal(7) * infoVal(9) * 9 + 1) 'Get 2nd data of 10th profile
```

3.5.6. CaoController::Execute("GetStorageBatchProfile") コマンド

ストレージの OUT 測定データを取得します.

書式 GetStorageBatchProfile (<dwSurface >,< dwGetBatchNo >,< dwGetBatchTopProfNo >, < byGetProfCnt >)

dwSurface	:	読み出しストレージ面 (VT_UI4)
dwGetBatchNo	:	読み出すバッチ No. (VT_UI4)
dwGetBatchTopProfNo	:	バッチ内の取得開始プロファイル No. (VT_UI4)
byGetProfCnt		読み出すプロファイル数 (VT_UI1)
戻り値	:	OUT 測定データ (VT_ARRAY|VT_VARIANT) データは表 3-11 のように格納されています.

表 3-11 GetStorageBatchProfile 取得データ

Index1	Index2	内容	データ型
0	0	ストレージの空き状態	VT_UI1
	1	対象ストレージ面のプログラム No.	VT_UI1
	2	ストレージ対象	VT_UI1
	3	ストレージ点数	VT_UI4
	4	今回読み出したバッチ No.	VT_UI4
	5	今回読み出したバッチ内のプロファイル数	VT_UI4
	6	読み出した中で最古のプロファイルが, バッチ内の何番目のプロファイルか	VT_UI4
	7	今回読み出したプロファイル数	VT_UI1
	8	基準時刻	VT_DATE
	9	1 単位のデータに格納されているプロファイル数	VT_UI1
	10	プロファイル圧縮(時間軸)の ON/OFF	VT_UI1
	11	1 プロファイル中のデータ点数	VT_UI2
	12	1 点目の X 座標	VT_I4
	13	データ点の X 方向の間隔	VT_I4
1	0~	プロファイルデータ 取得したプロファイルデータが順に格納されています. 1 単位のデータ数及びデータ格納順については「LJ-V7000 シリーズ 通信ライブラリ リファレンスマニュアル」9.2.9.6, 9.2.9.7 項を参照してください.	VT_UI4
2	0~	各プロファイルに対する OUT 測定結果	VT_UI1 または

		下記の要素数のデータが格納されています <16OUT 分のデータ(表 3-4 と同様のデータ順)> ×<読み出したデータ点数>	VT_R4
3	-	各データの基準時刻から 10ms 単位でカウントされた値	VT_UI4
4	0~47	バッチに対する OUT 測定結果 表 3-4 と同様のデータ順で格納されています	VT_UI1 または VT_R4

使用例

```
' 使用例プログラム
Dim retVal As Variant
Dim infoVal As Variant
Dim profVal As Variant
Dim data As Integer
retVal = caoCtrl.Execute("GetStorageProfile", Array(0, 0, 10))
infoVal = retVal(0)
profVal = retVal(1)
data = profVal(infoVal(9) * infoVal(11) * 9 + 1) 'Get 2nd data of 10th profile
```

3.6. 高速データ通信関連

3.6.1. CaoController::Execute("StartHighSpeedDataCommunication") コマンド

高速データ通信を開始します. このコマンドは, LJ-V7000 通信ライブラリの下記の 3 つのコマンドを実行します.

1. HighSpeedDataEthernetCommunicationInitialize
2. PreStartHighSpeedDataCommunication
3. StartHighSpeedDataCommunication

書式　StartHighSpeedDataCommunication (<wHighSpeedPortNo >,< dwProfileCnt >)

wHighSpeedPortNo	:	高速データ通信に用いるポート番号 (VT_UI2)
dwProfileCnt	:	まとめて送信するプロファイル数 (VT_UI4) ここで指定した数だけプロファイルを受信すると OnMessage イベントが発生します.
戻り値	:	なし

使用例

```
' 使用例プログラム
caoCtrl.Execute("StartHighSpeedDataCommunication", Array(24692, 10))
```

3.6.2. CaoController::Execute("StopHighSpeedDataCommunication") コマンド

高速データ通信を開始します. このコマンドは, LJ-V7000 通信ライブラリの下記の 2 つのコマンドを実行します.

1. StopHighSpeedDataCommunication
2. HighSpeedDataCommunicationFinalize

書式　StopHighSpeedDataCommunication ( )

引数　：　なし

戻り値　：　なし

使用例

```
' 使用例プログラム
caoCtrl.Execute "StopHighSpeedDataCommunication"
```