KEYENCE
LK-G5000 プロバイダ

Version 1.0.0

ユーザーズ ガイド

June 26, 2019

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	2019-06-26	初版.

【 動作確認機種 】

機種	バージョン	注意事項
LK-G5000		

目次

1. はじめに

本書は KEYENCE 社製超高速・高精度レーザ変位計 LK-G5000 シリーズの CAO プロバイダである，LK-G5000 プロバイダのユーザーズガイドです．

LK-G5000 プロバイダは Ethernet 接続された LK-G5000 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 LK-G5000 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

LK-G5000 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 LK-G5000 プロバイダ

ファイル名	CaoProvKEYENCELKG5000.dll
ProgID	CaoProv.KEYENCE.LK-G5000
レジストリ登録	regsvr32 CaoProvKEYENCELKG5000.dll
レジストリ登録の抹消	regsvr32 /u CaoProvKEYENCELKG5000.dll

表 2-2 依存モジュール

No.	ファイル名	説明
1	LkIF2.DLL	DLL 本体です
2	CmnLib.dll	DLL の動作に必要です
3	KeyUsbDrv.dll	DLL の動作に必要です

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.

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
bstrProvName	:	[in]	プロバイダ名. 固定値 ="CaoProv.KEYENCE.LK-G5000".
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 2-3 CaoWorkspace::AddController のオプション文字列

オプション	説明
IPAddress=<IP アドレス>	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. 例:IPAddress=192.168.0.2

使用例

```
// 使用例プログラム
Dim caoCtrl As Object
set caoCtrl = caoWs.AddController("LKG", "CaoProv.KEYENCE.LK-G5000", "",
"IPAddress=192.168.0.2")
```

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. エラーコード

LK-G5000 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 2-4 固有エラーコード

エラー番号	説明
0x80101001	コマンドエラー
0x80101002	コマンド長エラー
0x80101003	タイムアウト
0x80101004	チェックサムエラー
0x80101005	状態エラー
0x80101006	その他エラー
0x80101007	パラメータエラー
0x80101008	使用ヘッド数，OUT 数オーバー
0x80101009	使用ヘッド数，OUT 数オーバー
0x8010100A	演算対象に指定できない OUT を指定
0x8010100B	演算対象に指定されている OUT が無い
0x8010100C	使用できないサンプリング周期
0x8010100D	メインユニット異常
0x8010100E	設定値異常
0x80102000	デバイスのオープンに失敗
0x80102001	デバイスがオープンされていない
0x80102002	コマンド送信エラー
0x80102003	レスポンス受信エラー
0x80102004	タイムアウト
0x80102005	データなし
0x80102006	空きメモリなし
0x80102007	断線の可能性あり
0x80102008	未定義エラー

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「LK-G5000 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.

3.1. システム制御

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

指定した1つの OUT 番号の測定値を取得します.

書式　GetCalcDataSingle (<ulOutNo>)

ulOutNo　:　[in] 取得したい OUT 番号 (VT_UI4)
0 – 11

戻り値　:　[out] 結果データ (VT_VARIANT | VT_ARRAY)

Index	内容
0	0 番目の OUT 番号 (VT_I4)
1	0 番目の Flag (VT_I4) 0 : 有効データ 1 : +レンジオーバー 2 : -レンジオーバー 3 : 判定待機 4 : アラーム 5 : 無効(エラーなど)
2	0 番目の測定値 (VT_R4)

使用例

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

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

複数の OUT 番号を値を取得します.

書式　GetCalcDataMulti (<ulOutNo>)

ulOutNo : [in] 取得したい OUT 番号の論理和 (VT_UI4)

0x0001, // OUT1
0x0002, // OUT2
0x0004, // OUT3
0x0008, // OUT4
0x0010, // OUT5
0x0020, // OUT6
0x0040, // OUT7
0x0080, // OUT8
0x0100, // OUT9
0x0200, // OUT10
0x0400, // OUT11
0x0800, // OUT12

戻り値 : [out] 結果データ (VT_VARIANT | VT_ARRAY)

Index	内容
0	0 番目の OUT 番号 (VT_I4)
1	0 番目の Flag (VT_I4) 0 : 有効データ 1 : +レンジオーバー 2 : -レンジオーバー 3 : 判定待機 4 : アラーム 5 : 無効(エラーなど)
2	0 番目の測定値 (VT_R4)
...	
0 + n*3	n 番目の OUT 番号 (VT_I4)
1 + n*3	n 番目の Flag (VT_I4)
2 + n*3	n 番目の測定値 (VT_R4)

使用例

```
// 使用例プログラム
Dim retVal As Variant
retVal = caoCtrl.Execute("GetCalcDataMulti", 5)     // OUT1 と OUT3
```

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

全ての OUT 番号を値を取得します.

書式　GetCalcDataALL()

引数　：　[in] なし

戻り値　：　GetCalcDataMulti

Index	内容
0	0 番目の OUT 番号 (VT_I4)
1	0 番目の Flag (VT_I4) 0 : 有効データ 1 : +レンジオーバー 2 : -レンジオーバー 3 : 判定待機 4 : アラーム 5 : 無効(エラーなど)
2	0 番目の測定値 (VT_R4)
...	
33	11 番目の OUT 番号 (VT_I4)
34	12 番目の Flag (VT_I4)
35	13 番目の測定値 (VT_R4)

使用例

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

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

指定した1つの OUT 番号に対してタイミングを ON/OFF します.

書式　SetTimingSingle (<ulOutNo>, <ucTiming>)

ulOutNo　：　[in] 取得したい OUT 番号 (VT_UI4)

0 – 11

ucTiming : [in] タイミング (VT_UI1)
0: OFF
1: ON

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetTimingSingle", Array(5, 0))
```

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

指定した複数の OUT 番号に対してタイミングを ON/OFF します.

書式 SetTimingMulti (<ulOutNo>, <ucTiming>)

ulOutNo : [in] 取得したい OUT 番号の論理和 (VT_UI4)
0x0001, // OUT1
0x0002, // OUT2
0x0004, // OUT3
0x0008, // OUT4
0x0010, // OUT5
0x0020, // OUT6
0x0040, // OUT7
0x0080, // OUT8
0x0100, // OUT9
0x0200, // OUT10
0x0400, // OUT11
0x0800, // OUT12

ucTiming : [in] タイミング (VT_UI1)
0: OFF
1: ON

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetTimingMulti", Array(5, 0))
```

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

同期設定された OUT のタイミングを ON/OFF します.

書式　SetTimingSync (<ucTiming>)

ucTiming	:	[in] タイミング (VT_UI1) 0: OFF 1: ON
戻り値	:	[out] なし

使用例

```
// 使用例プログラム
call ctrl.Execute("SetTimingSync")
```

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

指定した1つの OUT 番号に対してオートゼロを ON/OFF します.

書式　SetTimingSingle (<ulOutNo>, <ucFlag>)

ulOutNo	:	[in] 取得したい OUT 番号 (VT_UI4) 0 – 11
ucFlag	:	[in] オートゼロの ON/OFF (VT_UI1) 0 : OFF 1 : ON
戻り値	:	[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetTimingSingle", Array(5,1))
```

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

指定した複数の OUT 番号のオートゼロを ON/OFF します.

書式　SetZeroMulti(<ulOutNo>)

ulOutNo	:	[in] 取得したい OUT 番号の論理和 (VT_UI4)
		0x0001, // OUT1
		0x0002, // OUT2
		0x0004, // OUT3
		0x0008, // OUT4
		0x0010, // OUT5
		0x0020, // OUT6
		0x0040, // OUT7
		0x0080, // OUT8
		0x0100, // OUT9
		0x0200, // OUT10
		0x0400, // OUT11
		0x0800, // OUT12
ucFlag	:	[in] オートゼロの ON/OFF (VT_UI1)
		0 : OFF
		1 : ON
戻り値	:	[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetZeroMulti", Array(5, 1))
```

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

同期設定された OUT のオートゼロを ON/OFF します.

書式　SetZeroSync (<ucFlag>)

ucFlag　：　[in] オートゼロの ON(1)／OFF(0) (VT_UI1)
0 : OFF
1 : ON

戻り値　：　[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetZeroSync", 0)
```

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

指定した複数の OUT 番号にリセットを入力します.

書式　SetResetMulti (<ulOutNo>,)

ulOutNo　：　[in] リセットしたい OUT 番号の論理和 (VT_UI4)
0x0001, // OUT1
0x0002, // OUT2
0x0004, // OUT3
0x0008, // OUT4
0x0010, // OUT5
0x0020, // OUT6
0x0040, // OUT7
0x0080, // OUT8
0x0100, // OUT9
0x0200, // OUT10
0x0400, // OUT11
0x0800, // OUT12

戻り値　：　[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetResetMulti", 5)
```

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

同期設定された OUT にリセットを入力します.

書式　SetResetSync ()

引数　：　[in] なし

戻り値　：　[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetResetSync")
```

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

パネルロックの ON/OFF を設定します.

書式　SetPanelLock (< ucFlag >)

ucFlag　：　[in] パネルロックの ON(1)/OFF(0) (VT_UI1)
0 : OFF
1 : ON

戻り値　：　[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetPanelLock", 0)
```

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

プログラム番号を指定した番号に切り替えます.

書式　SetProgramNo (< ucFlag >)

ucFlag　：　[in] プログラム番号 (VT_UI4)
0 - 7

戻り値　：　[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetProgramNo", 0)
```

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

現在のプログラム番号を取得します.

書式　GetProgramNo ( )

引数　：　[in] なし

戻り値　：　[out] プログラム番号 (VT_UI4)
0 - 7

使用例

```
// 使用例プログラム
Dim PrgNo as Variant
PrgNo = caoCtrl.Execute("GetProgramNo")
```

3.1.15. CaoController::Execute("DataStorageStart") コマンド

データストレージを開始します.

書式　DataStorageStart ()

引数　：　[in] なし

戻り値　：　[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("DataStorageStart")
```

3.1.16. CaoController::Execute("DataStorageStop") コマンド

データストレージを停止します.

書式　DataStorageStop ()

引数　:　[in] なし

戻り値　:　[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("DataStorageStop", 0)
```

3.1.17. CaoController::Execute("DataStorageGetData") コマンド

データストレージの蓄積データを取得します.

書式　DataStorageGetData (< ulOutNo >, <ulNum>)

ulOutNo　:　[in] 取得するデータの OUT 番号 (VT_UI4)

0 – 11

ulNum　[in] 取得するデータの個数

1 - 1200000

戻り値　:　[out] 取得したデータ (VT_VARIANT | VT_ARRAY)

Index	内容
0	0 番目の Flag (VT_I4) 0 : 有効データ 1 : +レンジオーバー 2 : -レンジオーバー 3 : 判定待機 4 : アラーム

	5 : 無効(エラーなど)
1	0 番目の測定値 (VT_R4)
...	
0 + n*2	n 番目の OUT 番号 (VT_I4)
1 + n*2	n 番目の Flag (VT_I4)

使用例

```
// 使用例プログラム
Dim vntRet as Variant
vntRet = caoCtrl.Execute("DataStorageGetData", 0)
```

3.1.18. CaoController::Execute("DataStorageInit") コマンド

データストレージの蓄積データをクリアします.

書式 DataStorageInit (< ucFlag >)

引数 : [in] なし

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("DataStorageInit")
```

3.1.19. CaoController::Execute("DataStorageGetStatus") コマンド

データストレージの蓄積状態を取得します.

書式 DataStorageGetStatus ()

引数 : [in] なし

戻り値 : [out] データストレージの蓄積状態 (VT_VARIANT | VT_ARRAY)

Index	内容
0	蓄積状態 (VT_I4) 0 : 停止中

	1：蓄積中
1	蓄積されているデータの数 (VT_I4)

使用例

```
// 使用例プログラム
Dim vntRet as Variant
vntRet = caoCtrl.Execute("DataStorageGetStatus")
```

3.1.20. CaoController::Execute("GetLight") コマンド

指定ヘッド番号の受光波形データを取得します.

書式　GetLight (<ulHeadNo>, <ulPeakNo>)

ulHeadNo　：　[in] 取得するデータのヘッド番号 (VT_UL4)
0 - 11

ulPeakNo　[in] 透明体2のとき, ピーク番号 (VT_UL4)
0 - 3

戻り値　：　[out] 受光波形データ(VT_VARIANT | VT_ARRAY)

Index	内容
0	ピーク番号 0 の測定位置 (VT_I4)
...	
3	ピーク番号 3 の測定位置 (VT_I4)
4	波形データ 0　(VT_UI1)
...	
2051	波形データ 2047 (VT_UI1)

使用例

```
// 使用例プログラム
Dim vntRet as Variant
vntRet = caoCtrl.Execute("GetLight", Array(5, 0))
```

3.2. 測定変更

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

表示パネルの上段・下段に表示する OUT 番号を設定します.

書式　SetPanel (<lUpperOUTNo>, < lLowerOUTNo >)

lUpperOUTNo	:	[in] 上段に表示する OUT 番号 (VT_I4)
		-1 : 非表示
		0 – 11 : OUT 番号
lLowerOUTNo		[in] 下段に表示する OUT 番号 (VT_I4)
		-1 : 非表示
		0 – 11 : OUT 番号
戻り値	:	[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetPanel", Array(5,7))
```

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

公差を設定します.

書式　SetTolerance (<lOUTNo>, < lUpperLimit>, <lLowerLimit>, <lToleranceHysteresis>)

lOUTNo	:	[in] OUT 番号 (VT_I4)
		0 – 11 : OUT 番号
lUpperLimit		[in] 公差上限値 (VT_I4)
		-999999 - 999999
lLowerLimit		[in] 公差下限値 (VT_I4)
		-999999 - 999999
lToleranceHysteresis		[in] 公差ヒステリシス (VT_I4)
		0 - 999999
戻り値	:	[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetTolerance", Array(5,-10, 100, 50))
```

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

ABLE モードを設定します.

書式 SetAbleMode (<ulHeadNo>, < ulABLEMode >)

ulHeadNo	:	[in] ヘッド番号 (VT_UI4)
		0 - 11
ulABLEMode		[in] ABLE モード (VT_UI4)
		0 : 自動
		1 : マニュアル
戻り値	:	[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetAbleMode", Array(0,0))
```

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

ABLE 制御範囲を設定します.

書式 SetAbleMinMax (< ulHeadNo >, < ulMinABLE >, <ulMaxABLE>)

ulHeadNo	:	[in] ヘッド番号 (VT_UI4)
		0 - 11
ulMinABLE		[in] ABLE 最小値 (VT_UI4)
		1-99
ulMaxABLE		[in] ABLE 最大値 (VT_UI4)
		1-99

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetAbleMinMax", Array(0, 1, 50))
```

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

測定モードを設定します.

書式 SetMeasureMode (< ulHeadNo >, < ulMode >)

ulHeadNo : [in] ヘッド番号 (VT_UI4)
0 - 11

ulMode [in] 測定モード (VT_UI4)
0 : 標準
1 : 半透明体
2 : 透明体
3 : 透明体 2
4 : 多重反射体｜光沢樹脂

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetMeasureMode", Array(0,0))
```

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

基点を設定します.

書式 SetBasicPoint (< ulHeadNo >, <ulPoint >)

ulHeadNo : [in] ヘッド番号 (VT_UI4)

0 - 11

ulPoint [in] 基点 (VT_UI4)

0 : NEAR

1 : FAR

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetBasicPoint", Array(0,0))
```

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

アラーム処理回数を設定します.

書式 SetNumAlarm (< ulHeadNo >, < ulCount >)

ulHeadNo : [in] ヘッド番号 (VT_UI4)

0 - 11

ulCount [in] アラーム処理回数 (VT_UI4)

0 - 9999

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetNumAlarm", Array(0,0))
```

3.2.8. CaoController::Execute("SetNumRecovery") コマンド

アラーム処理回数を設定します.

書式 SetNumRecovery (< ulHeadNo >, < ulCount >)

ulHeadNo : [in] ヘッド番号 (VT_UI4)

0 - 11

ulCount [in] アラーム復帰回数 (VT_UI4)

0 - 9999

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetNumRecovery ", Array(0,0))
```

3.2.9. CaoController::Execute("SetAlarmLevel ") コマンド

アラーム処理回数を設定します.

書式 SetAlarmLevel (< ulHeadNo >, < ulLevel >)

ulHeadNo : [in] ヘッド番号 (VT_UI4)

0 - 11

ulLevel [in] アラームレベル(VT_UI4)

0 - 9

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetAlarmLevel", Array(0,0))
```

3.2.10. CaoController::Execute("AbleStart") コマンド

ABLE チューニングを開始します.

書式 AbleStart (< ulHeadNo >, < ulCount >)

ulHeadNo : [in] ヘッド番号 (VT_UI4)

0 - 11

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("AbleStart", 0)
```

3.2.11. CaoController::Execute("AbleStop") コマンド

ABLE チューニング終了.

書式 AbleStop ()

引数 : [in] なし

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("AbleStop")
```

3.2.12. CaoController::Execute("AbleCancel") コマンド

ABLE チューニング中止.

書式 AbleCancel ()

引数 : [in] なし

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("AbleCancel")
```

3.2.13. CaoController::Execute("SetReflectionMode") コマンド

設置モードを設定します.

書式　SetReflectionMode (< ulHeadNo >, < ulMode >)

ulHeadNo	:	[in] ヘッド番号 (VT_UI4)
		0 - 11
ulMode		[in] 設置モード (VT_UI4)
		0 : 拡散反射
		1 : 正反射
戻り値	:	[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetReflectionMode", Array(0,0))
```

3.2.14. CaoController::Execute("SetMask") コマンド

マスクを設定します.

書式 SetMask (< ulHeadNo >, <bFlag>, <lMask1>, < lMask2>)

ulHeadNo	:	[in] ヘッド番号 (VT_UI4) 0 - 11
bFlag		[in] マスクの ON/OFF (VT_BOOL) FALSE : OFF TRUE : ON
lMask1		[in] マスク境界値 1 (VT_I4) -999999 - 999999
lMask2		[in] マスク境界値 2 (VT_I4) -999999 - 999999
戻り値	:	[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetMask", Array(0,0,0,0,0))
```

3.2.15. CaoController::Execute("SetMedian") コマンド

メディアンを設定します.

書式 SetMedian (< ulHeadNo >, < ulMedian >)

ulHeadNo	:	[in] ヘッド番号 (VT_UI4) 0 - 11
ulMedian		[in] メディアン (VT_UI4) 0 : OFF 1 : 7 点 2 : 15 点 3 : 31 点

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetMedian", Array(0,0))
```

3.2.16. CaoController::Execute("SetLaserCtrlGroup") コマンド

LASER CTRL グループを設定します.

書式 SetLaserCtrlGroup (< ulHeadNo >, < ulGroup >)

ulHeadNo : [in] ヘッド番号 (VT_UI4)
0 - 11
ulGroup [in] LASER CTRL グループ(VT_UI4)
0 : LASER CTRL 1
1 : LASER CTRL 2
戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetLaserCtrlGroup", Array(0,0))
```

3.2.17. CaoController::Execute("SetRange") コマンド

レンジを設定します.

書式 SetRange (< ulHeadNo >, < ulRange >)

ulHeadNo : [in] ヘッド番号 (VT_UI4)
0 - 11
ulRange [in] メディアン (VT_UI4)
0 : CENTER

1 : FAR

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetRange", Array(0,0))
```

3.2.18. CaoController::Execute("SetMutualInterferencePreventionGroup") コマンド

相互干渉防止のグループを設定します.

書式 SetMutualInterferencePreventionGroup(< ulHeadNo >, < ulGroup >)

ulHeadNo : [in] ヘッド番号 (VT_UI4)

0 - 11

ulGroup [in] 相互干渉防止のグループ (VT_UI4)

0 : Group A

1 : Group B

2 : Group C

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetMutualInterferencePreventionGroup", Array(0,0))
```

3.2.19. CaoController::Execute("SetCalcMethod") コマンド

メディアンを設定します.

書式 SetCalcMethod (< ulHeadNo >, < ulMethod >, <ulNo>)

ulHeadNo : [in] ヘッド番号 (VT_UI4)

0 - 11

ulMethod		[in] 演算方式 (VT_UI4)
		0 : ヘッド
		1 : OUT
		2 : 加算
		3 : 減算
		4 : AVE
		5 : P-P
		6 : MAX
		7 : MIN
ulNo		[in] 演算方式が HEAD, OUT の場合の番号 (VT_UI4)
戻り値	:	[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetCalcMethod", Array(0,0,0))
```

3.2.20. CaoController::Execute("SetCalcTarget") コマンド

測定面を設定します.

書式 SetCalcTarget (< ulHeadNo >, < ulTarget >)

ulHeadNo	:	[in] ヘッド番号 (VT_UI4)
		0 - 11
ulTarget		[in] 演算方式 (VT_UI4)
		0 : ピーク 1
		1 : ピーク 2
		2 : ピーク 3
		3 : ピーク 4
		4 : ピーク 1－ピーク 2
		5 : ピーク 1－ピーク 3
		6 : ピーク 1－ピーク 4
		7 : ピーク 2－ピーク 3
		8 : ピーク 2－ピーク 4

9 : ピーク 3－ピーク 4

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetCalcTarget", Array(0,0))
```

3.2.21. CaoController::Execute("SetOutAddSub") コマンド

OUT 間演算加減算の対象となる OUT 番号を設定します.

書式 SetOutAddSub(< ulHeadNo >, < ulOUT1 >, < ulOUT2 >)

ulHeadNo : [in] ヘッド番号 (VT_UI4)
0 - 11
ulOUT1 [in] 被加算(被減算)側 OUT 番号 (VT_UI4)
0 - 11
ulOUT2 [in] 加算(減算)側 OUT 番号 (VT_UI4)
0 - 11
戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetOutAddSub", Array(0,0,1))
```

3.2.22. CaoController::Execute("SetOutOperation") コマンド

OUT 間演算多項算(AVE/P-P/MAX/MIN)の対象となる OUT 番号を設定します.

書式 SetOutOperation (< ulOUTNo >, < ulOUTNo2>)

ulOUTNo : [in] OUT 番号(VT_UI4)
0 - 11

ulOUTNo2		[in] 対象とする OUT 番号 (VT_UI4) 複数指定は LKIF_OUTNO メンバの OR をとります. 0x0001, //OUT1 0x0002, //OUT2 0x0004, //OUT3 0x0008, //OUT4 0x0010, //OUT5 0x0020, //OUT6 0x0040, //OUT7 0x0080, //OUT8 0x0100, //OUT9 0x0200, //OUT10 0x0400, //OUT11 0x0800, //OUT12 0x0FFF, //全 OUT
戻り値	:	[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetOutOperation", Array(0,1))
```

3.2.23. CaoController::Execute("SetScaling") コマンド

スケーリングを設定します.

書式 SetScaling (< lOUTNo >, < lInput1>, < lOutput1>, < lInput2>, < lOutput2>)

lOUTNo	:	[in] 設定する OUT 番号 (VT_UI4) 0 - 11
lInput1		[in] 入力値 1 (VT_I4) -999999 - 999999
lOutput1		[in] 表示値 1 (VT_I4) -999999 - 999999
lInput2		[in] 入力値 2(VT_I4)

-999999 - 999999

lOutput2 [in] 表示値 2 (VT_I4)

-999999 - 999999

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetScaling", Array(0, 0, 0, 0, 0))
```

3.2.24. CaoController::Execute("SetFilter") コマンド

フィルタモード, フィルタパラメータを設定します.

書式 SetFilter (< ulOUTNo >, < ulMode >, <ulParam>)

ulOUTNo : [in] OUT 番号 (VT_UI4)

0 - 11

ulMode [in] フィルタモードを指定します (VT_UI4)

0 : 移動平均

1 : ローパスフィルタ

2 : ハイパスフィルタ

ulParam [in] フィルタパラメータ (VT_UI4)

(移動平均なら平均回数, LPF・HPF ならカットオフ周波数)

移動平均の場合

0 : 1 回

1 : 4

2 : 16

3 : 64

4 : 256

5 : 1024

6 : 4096

7 : 16384

8 : 65536

9 : 262144

LPF・HPF の場合

0 : 3000 Hz

1 : 1000

2 : 300

3 : 100

4 : 30

5 : 10

6 : 3

7 : 1

8 : 0.3

9 : 0.1

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetFilter", Array(0,0, 0))
```

3.2.25. CaoController::Execute("SetTriggerMode") コマンド

トリガモードを設定します.

書式 SetTriggerMode (< ulOUTNo >, < ulMode >)

ulOUTNo : [in] OUT 番号 (VT_UI4)

0 - 11

ulMode [in] トリガモード (VT_UI4)

0 : 外部トリガ 1

1 : 外部トリガ 2

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetTriggerMode", Array(0,0))
```

3.2.26. CaoController::Execute("SetOffset") コマンド

オフセットを設定します.

書式　SetOffset (< lOUTNo >, < lOffset >)

lOUTNo	:	[in] OUT 番号 (VT_I4)
		0 - 11
lOffset		[in] オフセット (VT_I4)
		-999999 - 999999
戻り値	:	[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetOffset", Array(0,0))
```

3.2.27. CaoController::Execute("SetCalcMode") コマンド

計測モードを設定します.

書式　SetCalcMode (< ulOUTNo >, < ulMode >)

ulOUTNo	:	[in] OUT 番号 (VT_UI4)
		0 - 11
ulMode		[in] トリガモード (VT_UI4)
		0 : 標準
		1 : ピークホールド
		2 : ボトムホールド
		3 : ピーク to ピークホールド
		4 : サンプルホールド
		5 : アベレージホールド
戻り値	:	[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetCalcMode", Array(0,0))
```

3.2.28. CaoController::Execute("SetAnalogScaling") コマンド

アナログ出力スケーリングを設定します.

書式 SetAnalogScaling (< lOUTNo >, < lInput1 >, <lOutput1>, < lInput2 >, <lOutput2>)

lOUTNo	:	[in] OUT 番号 (VT_I4) 0 - 11
lInput1	:	[in] 入力値1(VT_I4) -999999 - 999999
lOutput1	:	[in] 出力電圧 1 (VT_I4) -10500(mV) – 10500(vm)
lInput2	:	[in] 入力値 2(VT_I4) -999999 - 999999
lOutput2	:	[in] 出力電圧 2 (VT_I4) -10500(mV) – 10500(vm)
戻り値	:	[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetAnalogScaling", Array(0, 0, 0, 100, 100))
```

3.2.29. CaoController::Execute("SetDisplayUnit") コマンド

最小表示単位を設定します.

書式 SetDisplayUnit (< ulOUTNo >, < ulMode >)

ulOUTNo	:	[in] OUT 番号 (VT_UI4)

0 - 11

ulMode [in] 表示単位 (VT_UI4)

0 : 0.01mm
1 : 0.001mm
2 : 0.0001mm
3 : 0.00001mm
4 : 0.1μm
5 : 0.01μm
6 : 0.001μm

0 : 0.1m/s
1 : 0.01m/s
2 : 0.001m/s
3 : 0.1mm/s
4 : 0.01mm/s
5 : 0.001mm/s
6 : 0.0001mm/s

0 : 0.1km/s2
1 : 0.01km/s2
2 : 0.001km/s2
3 : 0.1m/s2
4 : 0.01m/s2
5 : 0.001m/s2
6 : 0.0001m/s2

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetDisplayUnit", Array(0,0))
```

3.2.30. CaoController::Execute("SetMeasureType") コマンド

測定種別を設定します.

書式　SetMeasureType (< ulOUTNo >, < ulMode >)

ulOUTNo	:	[in] OUT 番号 (VT_UI4)
		0 - 11
ulMode		[in] 測定種別 (VT_UI4)
		0 : 変位
		1 : 速度
		1 : 加速度
戻り値	:	[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetMeasureType", Array(0,0))
```

3.2.31. CaoController::Execute("SetSynchronization") コマンド

タイミング同期を設定します.

書式　SetSynchronization (< ulOUTNo >, < ulMode >)

ulOUTNo	:	[in] OUT 番号 (VT_UI4)
		0 - 11
ulMode		[in] 蓄積対象とするかどうか (VT_UI4)
		0 : 対象外
		1 : 対象
戻り値	:	[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetSynchronization", Array(0,0))
```

3.2.32. CaoController::Execute("SetStorageTarget") コマンド

ストレージの蓄積対象を設定します.

書式　SetStorageTarget (< ulOUTNo >, < ulMode >)

ulOUTNo	:	[in] OUT 番号 (VT_UI4)
		0 - 11
ulMode		[in] 蓄積対象とするかどうか (VT_UI4)
		0 : 対象外
		1 : 対象
戻り値	:	[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetStorageTarget", Array(0,0))
```

3.2.33. CaoController::Execute("SetSamplingCycle") コマンド

サンプリング周期を設定します.

書式　SetSamplingCycle (< ulCycle >)

ulCycle	[in] サンプリング周期(VT_UI4)
	0 : 2.55us
	1 : 5us
	2 : 10us
	3 : 20us
	4 : 50us
	5 : 100us
	6 : 200us
	7 : 500us
	8 : 1ms

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetSamplingCycle", 3)
```

3.2.34. CaoController::Execute("SetMutualInterferencePrevention") コマンド

相互干渉防止を設定します.

書式 SetMutualInterferencePrevention (< ulMode >)

ulMode [in] 相互干渉防止の指定(VT_UI4)
0 : OFF
1 : AB-ON
2 : ABC-ON

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetMutualInterferencePrevention", 1)
```

3.2.35. CaoController::Execute("SetToleranceComparatorOutputFormat") コマンド

サンプリング周期を設定します.

書式 SetToleranceComparatorOutputFormat (< ulMode >)

ulMode [in] 判定出力形態モード(VT_UI4)
0 : ノーマル
1 : ホールド
2 : オフディレイ

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetToleranceComparatorOutputFormat", 1)
```

3.2.36. CaoController::Execute("SetStrobeTime") コマンド

ストローブ時間を設定します.

書式 SetStrobeTime (< ulCycle >)

ulCycle		[in] ストローブ時間(VT_UI4)
		0 : 2ms
		1 : 5ms
		2 : 10ms
		3 : 20ms
戻り値	:	[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetStrobeTime", 3)
```

3.2.37. CaoController::Execute("SetDataStorage") コマンド

データストレージ(蓄積数, 蓄積周期)を設定します.

書式 SetDataStorage (<ulPoint>, < ulCycle >)

ulPoint	:	[in] 蓄積点数(VT_UI4)
		最大数は蓄積対象数で変わります
		1 – 1200000
ulCycle	:	[in] 蓄積周期 (VT_UI4)
		0 : サンプリング周期×1
		1 : サンプリング周期×2

2 : サンプリング周期×5
3 : サンプリング周期×10
4 : サンプリング周期×20
5 : サンプリング周期×50
6 : サンプリング周期×100
7 : サンプリング周期×200
8 : サンプリング周期×500
9 : サンプリング周期×1000
10 : タイミング同期

戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetDataStorage", Array(1000, 10))
```

3.2.38. CaoController::Execute("SetAnalogChannel") コマンド

アナログ出力 Ch を設定します.

書式 SetAnalogChannel (<ulChNo>, < ulOUTNo >)

ulChNo : [in] アナログ出力 Ch 番号(VT_UI4)
0 - (使用アナログ出力 Ch 数-1)
ulOUTNo [in] 出力する OUT 番号 (VT_UI4)
0 - 11
戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetAnalogChannel", Array(0,0))
```

3.2.39. CaoController::Execute("SetAlarmOutputForm") コマンド

アラーム出力形態を設定します.

書式　SetAlarmOutputForm (< ulMode >)

ulMode	:	[in] アラーム出力形態(VT_UI4) 0 : システムアラームのみ 1 : 測定値アラームのみ 2 : システムアラーム及び測定値アラーム
戻り値	:	[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetAlarmOutputForm", 2)
```

3.2.40. CaoController::Execute("SetNumOfUsedHeads") コマンド

使用ヘッド数を設定します.

書式　SetNumOfUsedHeads (< ulHeadNum >)

ulHeadNum	:	[in] 使用ヘッド数 (VT_UI4) 2 - 12
戻り値	:	[out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetNumOfUsedHeads", 2)
```

3.2.41. CaoController::Execute("SetNumOfUsedOut") コマンド

使用 OUT 数を設定します.

書式　SetNumOfUsedOut (< ulOUTNum >)

ulOUTNum : [in] 使用 OUT 数 (VT_UI4)
2 - 12
戻り値 : [out] なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("SetNumOfUsedOut", 2)
```

3.2.42. CaoController::Execute("GetPanel") コマンド

表示パネルの上段・下段に表示する OUT 番号を取得します.

書式 GetPanel ()

引数 : [in] なし
戻り値 : [out] 設定番号 (VT_VARIANT | VT_ARRAY)

Index	内容
0	上段に表示する OUT 番号 (VT_I4)
1	下段に表示する OUT 番号 (VT_I4)

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetPanel")
```

3.2.43. CaoController::Execute("GetTolerance") コマンド

公差を取得します.

書式 GetTolerance (<ulOUTNo>)

ulOUTNo : [in] OUT 番号 (VT_UI4)
0 - 11
戻り値 : [out] 設定番号 (VT_VARIANT | VT_ARRAY)

Index	内容
0	公差上限値 (VT_I4)
1	公差下限値 (VT_I4)
2	公差ヒステリシスの値 (VT_I4)

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetTolerance", 0)
```

3.2.44. CaoController::Execute("GetAbleMode") コマンド

ABLE モードを取得します.

書式 GetAbleMode (<ulOUTNo>)

ulOUTNo : [in] OUT 番号 (VT_UI4)
0 - 11

戻り値 : [out] ABLE モード (VT_I4)
0 : 自動
1 : マニュアル

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetAbleMode",1)
```

3.2.45. CaoController::Execute("GetAbleMinMax") コマンド

ABLE 制御範囲を取得します.

書式 GetAbleMinMax (<ulOUTNo>)

ulOUTNo : [in] OUT 番号 (VT_UI4)

0 - 11

戻り値 : [out] ABLE 制御範囲 (VT_VARIANT | VT_ARRAY)

Index	内容
0	ABLE 最小値 (VT_I4)
1	ABLE 最大値 (VT_I4)

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetAbleMinMax", 1)
```

3.2.46. CaoController::Execute("GetMeasureMode") コマンド

測定モードを取得します.

書式 GetMeasureMode (<ulHeadNo>)

ulHeadNo : [in] ヘッド番号 (VT_UI4)

0 - 11

戻り値 : [out] 測定モード (VT_I4)

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetMeasureMode", 1)
```

3.2.47. CaoController::Execute("GetBasicPoint") コマンド

基点を取得します.

書式 GetBasicPoint (<ulHeadNo>)

ulHeadNo : [in] ヘッド番号 (VT_UI4)

0 - 11

戻り値 : [out] 基点 (VT_I4)
0 : NEAR
1 : FAR

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetBasicPoint", 1)
```

3.2.48. CaoController::Execute("GetNumAlarm") コマンド

アラーム処理回数を取得します.

書式 GetNumAlarm (<ulHeadNo>)

ulHeadNo : [in] ヘッド番号 (VT_UI4)
0 - 11
戻り値 : [out] アラーム処理回数 (VT_I4)
0 - 9999

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetNumAlarm", 1)
```

3.2.49. CaoController::Execute("GetNumRecovery") コマンド

アラーム復帰回数を取得します.

書式 GetNumRecovery (<ulHeadNo>)

ulHeadNo : [in] ヘッド番号 (VT_UI4)
0 - 11
戻り値 : [out] アラーム復帰回数 (VT_I4)

0 - 9999

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetNumRecovery", 1)
```

3.2.50. CaoController::Execute("GetAlarmLevel") コマンド

アラームレベルを取得します.

書式　GetAlarmLevel (<ulHeadNo>)

ulHeadNo	:	[in] ヘッド番号 (VT_UI4)
		0 - 11
戻り値	:	[out] アラームレベル (VT_I4)
		0 - 9

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetAlarmLevel", 1)
```

3.2.51. CaoController::Execute("GetReflectionMode") コマンド

設置モードを取得します.

書式　GetReflectionMode (<ulHeadNo>)

ulHeadNo	:	[in] ヘッド番号 (VT_UI4)
		0 - 11
戻り値	:	[out] 設置モード (VT_I4)
		0 : 拡散反射
		1 : 正反射

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetReflectionMode", 1)
```

3.2.52. CaoController::Execute("GetMask") コマンド

マスクを取得します.

書式　GetMask (<ulHeadNo>)

ulHeadNo　:　[in] ヘッド番号 (VT_UI4)

0 - 11

戻り値　:　[out] マスク設定 (VT_VARIANT | VT_ARRAY)

Index	内容
0	マスクの ON/OFF (VT_I4) 0 : OFF 1 : ON
1	マスク境界値 1 (VT_I4) -999999 - 999999
2	マスク境界値 2 (VT_I4) -999999 - 999999

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetMask", 1)
```

3.2.53. CaoController::Execute("GetMedian") コマンド

メディアンを取得します.

書式　GetMedian (<ulHeadNo>)

ulHeadNo : [in] ヘッド番号 (VT_UI4)
0 - 11
戻り値 : [out] メディアン (VT_I4)
0 : OFF
1 : 7 点
2 : 15 点
3 : 31 点

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetMedian", 1)
```

3.2.54. CaoController::Execute("GetLaserCtrlGroup") コマンド

LASER CTRL グループを取得します.

書式 GetLaserCtrlGroup (<ulHeadNo>)

ulHeadNo : [in] ヘッド番号 (VT_UI4)
0 - 11
戻り値 : [out] LASER CTRL グループ(VT_I4)
0 : LASER CTRL 1
1 : LASER CTRL 2

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetLaserCtrlGroup", 1)
```

3.2.55. CaoController::Execute("GetRange") コマンド

メディアンを取得します.

書式　GetRange (<ulHeadNo>)

ulHeadNo	:	[in] ヘッド番号 (VT_UI4)
		0 - 11
戻り値	:	[out] レンジ (VT_I4)
		0 : CENTER
		1 : FAR

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetRange", 1)
```

3.2.56. CaoController::Execute("GetMutualInterferencePreventionGroup") コマンド

相互干渉防止のグループを取得します.

書式　GetMutualInterferencePreventionGroup (<ulHeadNo>)

ulHeadNo	:	[in] ヘッド番号 (VT_UI4)
		0 - 11
戻り値	:	[out] 相互干渉防止のグループ (VT_I4)
		0 : Group A
		1 : Group B
		2 : Group C

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetMutualInterferencePreventionGroup", 1)
```

3.2.57. CaoController::Execute("GetCalcTarget") コマンド

測定面を取得します.

書式　GetCalcTarget (<ulHeadNo>)

ulHeadNo　：　[in] ヘッド番号 (VT_UI4)
0 - 11

戻り値　：　[out] 演算方式 (VT_I4)
0：ピーク 1
1：ピーク 2
2：ピーク 3
3：ピーク 4
4：ピーク 1－ピーク 2
5：ピーク 1－ピーク 3
6：ピーク 1－ピーク 4
7：ピーク 2－ピーク 3
8：ピーク 2－ピーク 4
9：ピーク 3－ピーク 4

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetCalcTarget", 1)
```

3.2.58. CaoController::Execute("GetOutAddSub") コマンド

OUT 間演算加減算の対象となる OUT 番号を取得します.

書式　GetOutAddSub (<ulOUTNo>)

ulOUTNo　：　[in] OUT 番号 (VT_UI4)
0 - 11

戻り値　：　[out] 演算対象(加減算) (VT_VARIANT | VT_ARRAY)

Index	内容
0	被加算(被減算)側 OUT 番号 (VT_I4)
1	加算(減算)側 OUT 番号 (VT_I4)

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetOutAddSub", 1)
```

3.2.59. CaoController::Execute("GetOutOperation") コマンド

OUT 間演算多項算(AVE/P-P/MAX/MIN)の対象となる OUT 番号を取得します.

書式 GetOutOperation (<ulOUTNo>)

ulOUTNo : [in] OUT 番号 (VT_UI4)
0 - 11

戻り値 : [out] 演算対象(加減算) (VT_I4)
対象とする OUT 番号が格納されます.
複数指定は LKIF_OUTNO メンバの OR となっています.
0x0001 : OUT1
0x0002 : OUT2
0x0004 : OUT3
0x0008 : OUT4
0x0010 : OUT5
0x0020 : OUT6
0x0040 : OUT7
0x0080 : OUT8
0x0100 : OUT9
0x0200 : OUT10
0x0400 : OUT11
0x0800 : OUT12
0x0FFF : 全 OUT

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetOutOperation", 1)
```

3.2.60. CaoController::Execute("GetScaling") コマンド

スケーリングを取得します.

書式　GetScaling (<ulOUTNo>)

ulOUTNo　:　[in] OUT 番号 (VT_UI4)

0 - 11

戻り値　:　[out] スケーリング (VT_VARIANT | VT_ARRAY)

Index	内容
0	入力値 1　(VT_I4)
1	表示値 1　(VT_I4)
2	入力値 2　(VT_I4)
3	表示値 2　(VT_I4)

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetScaling", 1)
```

3.2.61. CaoController::Execute("GetFilter") コマンド

フィルタモード, フィルタパラメータを取得します.

書式　GetFilter (<ulOUTNo>)

ulOUTNo　:　[in] OUT 番号 (VT_UI4)

0 - 11

戻り値　:　[out] フィルタモード, フィルタパラメータ (VT_VARIANT |VT_ARRAY)

Index	内容
0	フィルタモード　(VT_I4) 0 : 移動平均

	1 : ローパスフィルタ 2 : ハイパスフィルタ
1	フィルタパラメータ (VT_I4) (移動平均なら平均回数, LPF・HPF ならカットオフ周波数) 移動平均の場合 0 : 1 回 1 : 4 2 : 16 3 : 64 4 : 256 5 : 1024 6 : 4096 7 : 16384 8 : 65536 9 : 262144 LPF・HPF の場合 0 : 3000 Hz 1 : 1000 2 : 300 3 : 100 4 : 30 5 : 10 6 : 3 7 : 1 8 : 0.3 9 : 0.1

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetFilter", 1)
```

3.2.62. CaoController::Execute("GetTriggerMode") コマンド

トリガモードを取得します.

書式　GetTriggerMode (<ulOUTNo>)

ulOUTNo	:	[in] OUT 番号 (VT_UI4) 0 - 11
戻り値	:	[out] トリガモード (VT_I4) 0 : 外部トリガ 1 1 : 外部トリガ 2

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetTriggerMode", 1)
```

3.2.63. CaoController::Execute("GetOffset") コマンド

オフセットを取得します.

書式　GetOffset (<ulOUTNo>)

ulOUTNo	:	[in] OUT 番号 (VT_UI4) 0 - 11
戻り値	:	[out] オフセット (VT_I4) -999999 - 999999

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetOffset", 1)
```

3.2.64. CaoController::Execute("GetCalcMode") コマンド

計測モードを取得します.

書式　GetCalcMode (<ulOUTNo>)

ulOUTNo　：　[in] OUT 番号 (VT_UI4)
0 - 11

戻り値　：　[out] 計測モード (VT_I4)
0：標準
1：ピークホールド
2：ボトムホールド
3：ピーク to ピークホールド
4：サンプルホールド
5：アベレージホールド

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetCalcMode", 1)
```

3.2.65. CaoController::Execute("GetAnalogScaling") コマンド

アナログ出力スケーリングを取得.

書式　GetAnalogScaling (<ulOUTNo>)

ulOUTNo　：　[in] OUT 番号 (VT_UI4)
0 - 11

戻り値　：　[out] アナログ出力スケーリング (VT_VARIANT | VT_ARRAY)

Index	内容
0	入力値 1　(VT_I4) -999999 - 999999
1	出力電圧 1　(VT_I4) -10500(mV) - 10500(mV)
2	入力値 2　(VT_I4) -999999 - 999999

3	出力電圧 2　(VT_I4) -10500(mV) - 10500(mV)

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetAnalogScaling", 1)
```

3.2.66. CaoController::Execute("GetDisplayUnit") コマンド

トリガモードを取得します.

書式　GetDisplayUnit (<ulOUTNo>)

ulOUTNo　:　[in] OUT 番号 (VT_UI4)
0 - 11

戻り値　:　[out] 最小表示単位 (VT_I4)
0 : 0.01mm
1 : 0.001mm
2 : 0.0001mm
3 : 0.00001mm
4 : 0.1μm
5 : 0.01μm
6 : 0.001μm

0 : 0.1m/s
1 : 0.01m/s
2 : 0.001m/s
3 : 0.1mm/s
4 : 0.01mm/s
5 : 0.001mm/s
6 : 0.0001mm/s

0 : 0.1km/s2

1 : 0.01km/s2

2 : 0.001km/s2

3 : 0.1m/s2

4 : 0.01m/s2

5 : 0.001m/s2

6 : 0.0001m/s2

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetDisplayUnit", 1)
```

3.2.67. CaoController::Execute("GetMeasureType") コマンド

測定種別を取得します.

書式　GetMeasureType (<ulOUTNo>)

ulOUTNo	:	[in] OUT 番号 (VT_UI4)
		0 - 11
戻り値	:	[out] 測定種別 (VT_I4)
		0 : 変位
		1 : 速度
		2 : 加速度

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetMeasureType", 1)
```

3.2.68. CaoController::Execute("GetSynchronization") コマンド

同期設定を取得します.

書式　GetSynchronization (<ulOUTNo>)

ulOUTNo	:	[in] OUT 番号 (VT_UI4)
		0 - 11
戻り値	:	[out] 蓄積対象とするかどうか (VT_I4)
		0 : 対象外
		1 : 対象

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetSynchronization", 1)
```

3.2.69. CaoController::Execute("GetStorageTarget") コマンド

ストレージの蓄積対象を取得します.

書式　GetStorageTarget (<ulOUTNo>)

ulOUTNo	:	[in] OUT 番号 (VT_UI4)
		0 - 11
戻り値	:	[out] 蓄積対象とするかどうか (VT_I4)
		0 : 対象外
		1 : 対象

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetStorageTarget", 1)
```

3.2.70. CaoController::Execute("GetSamplingCycle") コマンド

サンプリング周期が格納されます.

書式　GetSamplingCycle ()

引数　：[in] なし

戻り値　：[out] 相互干渉防止のグループ (VT_I4)

0 : 2.55us

1 : 5us

2 : 10us

3 : 20us

4 : 50us

5 : 100us

6 : 200us

7 : 500us

8 : 1ms

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetSamplingCycle")
```

3.2.71. CaoController::Execute("GetMutualInterferencePrevention") コマンド

相互干渉防止を取得します.

書式　GetMutualInterferencePrevention ()

引数　：[in] なし

戻り値　：[out] 相互干渉防止(VT_I4)

0 : OFF

1 : AB-ON

2 : ABC-ON

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetMutualInterferencePrevention")
```

3.2.72. CaoController::Execute("GetToleranceComparatorOutputFormat") コマンド

相互干渉防止を取得します.

書式　GetToleranceComparatorOutputFormat ()

引数　：[in] なし

戻り値　：[out] 判定出力形態モード(VT_I4)

0：ノーマル

1：ホールド

2：オフディレイ

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetToleranceComparatorOutputFormat")
```

3.2.73. CaoController::Execute("GetDataStorage") コマンド

データストレージ(蓄積数, 蓄積周期)を取得します.

書式　GetDataStorage ()

引数　：[in] なし

戻り値　：[out] データストレージ （VT_VARIANT | VT_ARRAY)

Index	内容
0	蓄積点数　(VT_I4) 1 - 1200000
1	蓄積周期　(VT_I4) 0：サンプリング周期×1 1：サンプリング周期×2 2：サンプリング周期×5 3：サンプリング周期×10

	4：サンプリング周期×20 5：サンプリング周期×50 6：サンプリング周期×100 7：サンプリング周期×200 8：サンプリング周期×500 9：サンプリング周期×1000

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetDataStorage")
```

3.2.74. CaoController::Execute("GetAnalogChannel") コマンド

アナログ出力 Ch を取得します.

書式　GetAnalogChannel (<ulChNo>)

ulChNo　：　[in] アナログ出力 Ch 番号 (VT_UI4)
戻り値　：　[out] 出力する OUT 番号 (VT_I4)

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetAnalogChannel", 1)
```

3.2.75. CaoController::Execute("GetAlarmOutputForm") コマンド

アラーム出力形態を取得します.

書式　GetAlarmOutputForm ()

引数　：　[in] なし
戻り値　：　[out] アラーム出力形態 (VT_I4)

0 : システムアラームのみ

1 : 測定値アラームのみ

2 : システムアラーム及び測定値アラーム

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetAlarmOutputForm")
```

3.2.76. CaoController::Execute("GetNumOfUsedHeads") コマンド

使用ヘッド数を取得します.

書式　GetNumOfUsedHeads ()

引数　：[in] なし

戻り値　：[out] 使用ヘッド数 （VT_I4)

2 - 12

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetNumOfUsedHeads")
```

3.2.77. CaoController::Execute("GetNumOfUsedOut") コマンド

使用 OUT 数を取得します.

書式　GetNumOfUsedOut ()

引数　：[in] なし

戻り値　：[out] 使用 OUT 数 （VT_I4)

2 - 12

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetNumOfUsedOut")
```

3.2.78. CaoController::Execute("GetNumOfUsedAnalogCh") コマンド

使用アナログ出力 Ch 数を取得します.

書式　GetNumOfUsedAnalogCh ()

引数　：　[in] なし

戻り値　：　[out] 使用アナログ出力 Ch 数 （VT_I4)

2 - 12

使用例

```
// 使用例プログラム
Dim vntRet
vntRet = caoCtrl.Execute("GetNumOfUsedAnalogCh")
```

3.3. モード変更コマンド

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

測定モードに遷移します.

書式　StartMeasure ( )

引数　：　なし

戻り値　：　なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("StartMeasure")
```

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

設定(通信)モードに遷移します.

書式　StopMeasure ( )

引数　：　なし

戻り値　：　なし

使用例

```
// 使用例プログラム
call caoCtrl.Execute("StopMeasure")
```