KEYENCE
LK-G3000LkIF プロバイダ
ユーザーズ ガイド

Version 1.0.0

June 10, 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	2020-06-10	初版.

【動作確認機種】

機種	バージョン	注意事項
LK-G3000	-	

【対応機種】

機種
LK-G3000
LK-G3000P
LK-G3000V
LK-G3000PV

目次

1. はじめに

本書は，KEYENCE 社製レーザー変位計 LK-G3000 シリーズに対して，KEYENCE 社製通信ライブラリ「LkIF.dll」を使用し，USB 通信にてデータのやり取りをするプロバイダのユーザーズガイドです．図 1-1 が本プロバイダとデバイスの全体構成図になります．以降本プロバイダを LK-G3000LkIF プロバイダと呼称します．

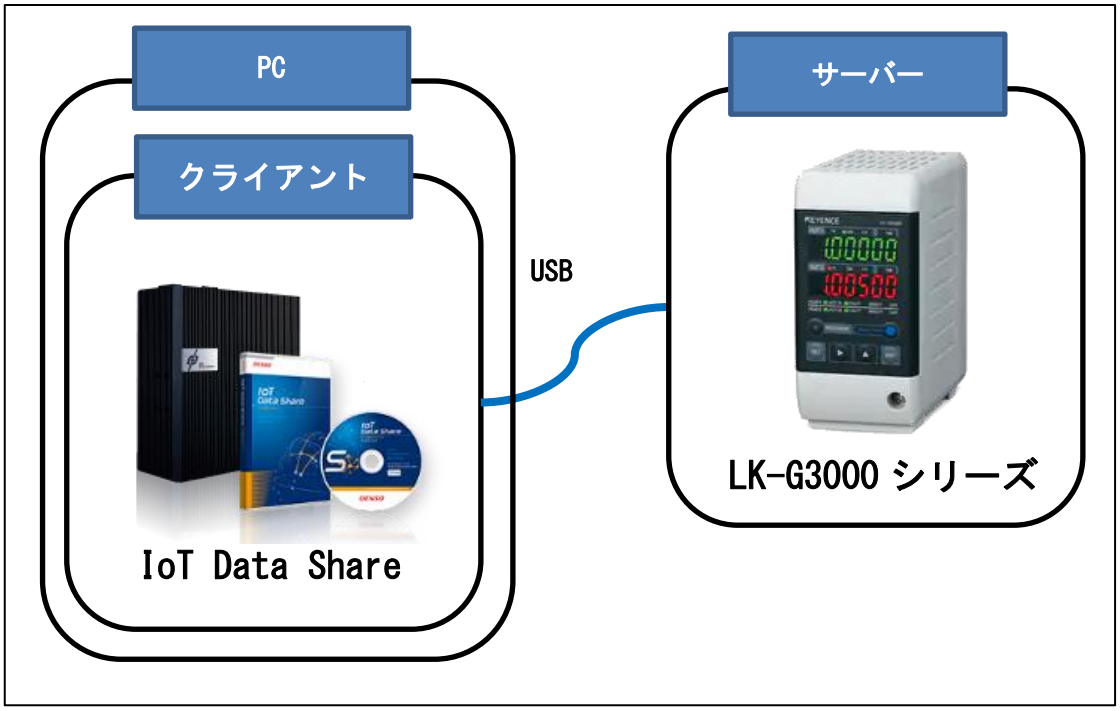


図 1-1 構成図

また，本プロバイダ及びデバイスそれぞれの対応を図 1-2に表します．
（※一例です．全てを表しているわけではありません．）

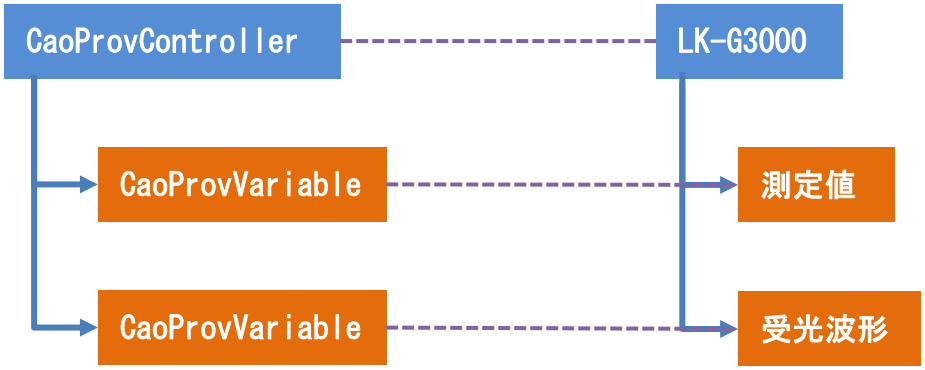


図 1-2 プロバイダの構成とデバイス情報との対応図

1.1. 参考となる情報源

LK-G3000LkIF プロバイダは，KEYENCE 社の「高速・高精度 CCD レーザー変位計 LK-G シリーズユーザーズマニュアル 96M12274」及び，「LK-G シリーズ用設定・支援ソフト LK-H1W LK-Navigator ユーザーズマニュアル 130074」を参考にしております．以降このマニュアルを，LK-G3000 マニュアル，LK-Navigator マニュアルと呼称します．

2. アプリケーション開発のための環境セットアップ

2.1. LK-G3000 とクライアント PC との接続

LK-G3000LkIF プロバイダは KEYENCE 社製通信ライブラリを使用して LK-G3000 シリーズと USB 通信を行います. USB 接続方法の詳細につきましては, LK-Navigator マニュアルを参照してください. また, 使用する KEYENCE 社製通信ライブラリは, 本プロバイダの dll と同じ Bin フォルダ内に同梱されております. 以下に, 使用する依存モジュールである KEYENCE 社製通信ライブラリの詳細を記述します.

表 2-1 依存モジュール

DLL	説明
LkIF.dll	DLL 本体です.
KeyUsbDrv.dll	DLL の動作に必要です.

2.2. PC 開発環境のセットアップ

初めて LK-G3000 シリーズとクライアント PC で USB 接続を行う際は, KEYENCE 社製の「LK-Navigator」の USB ドライバをインストールしてください. インストール方法の詳細につきましては, LK-Navigator マニュアルをご確認ください.

2.2.1. LK-G3000LkIF プロバイダの手動インストール

LK-G3000LkIF プロバイダを手動でインストールする場合は下記レジストリ登録を行う必要があります. レジストリ登録を行う場合は, 管理者権限でコマンドプロンプトを起動し, regsvr32 コマンドを実行してください. 実行する際には, ファイルのあるパスまで移動するか, ファイルパスを指定して実行してください.

表 2-2 LK-G3000LkIF プロバイダ

ファイル名	CaoProvKEYENCELK-G3000LkIF.dll
ProgID	CaoProv.KEYENCE.LK-G3000LkIF
レジストリ登録	regsvr32 CaoProvKEYENCELK-G3000LkIF.dll
レジストリ登録の抹消	regsvr32 /u CaoProvKEYENCELK-G3000LkIF.dll

3. コマンドリファレンス

3.1. メソッド/プロパティ一覧

表 3-1 メソッド/プロパティ一覧

カテゴリ	メソッド/プロパティ[1]		機能	参照
CaoWorkspace				
	AddController	M	コントローラに接続	P.10
CaoController				
	VariableNames	P	接続可能な変数名リストの取得	P.11
	Variables	P	コントローラが保持する変数コレクションの取得	P.11
	AddVariable	M	変数オブジェクトの追加	P.12
	Execute	M	拡張コマンドの実行	P.12
CaoVariable				
	Value	P	値の取得/設定	P.12

3.2. メソッド・プロパティ

3.2.1. CaoWorkspace クラス

3.2.1.1. AddController メソッド

CaoWorkspace に，コントローラオブジェクトを追加します．以下に，AddController メソッドの仕様を示します．

書式

```
AddController
(
        "<コントローラ名>",                    // コントローラ名(任意)
        "CaoProv.KEYENCE.LK-G3000LkIF",        // プロバイダ名 (固定)
        "<マシン名>",                          // プロバイダ実行マシン名(未使用)
        "<オプション>"                         // オプション文字列(省略可能)
)
```

オプション

以下にオプション文字列に指定するオプションを示します．オプション文字列は下記に示す各オプ

[1] M:メソッド，P:プロパティ，E:イベントをそれぞれ示します．

ションをカンマ(,)でつなげた文字列となります.

オプション	必須	説明	値範囲	デフォルト値
Timeout	--	応答待機時間	0 - 2147483647	500

使用例

```
Dim caoEng As CaoEngine          ' Engineオブジェクト
Dim caoWs As CaoWorkspace        ' Workspaceオブジェクト
Dim caoCtrl As CaoController     ' Controllerオブジェクト

' CaoEngine オブジェクトの生成
Set caoEng  = new CaoEngine
' CaoWorkspace オブジェクトの生成
Set caoWs   = caoEng.Workspaces.Item(0)
' CaoController オブジェクトの生成
Set caoCtrl = caoWs.AddController("LKG3000LkIF", _
                                  "CaoProv.KEYENCE.LK-G3000LkIF", _
                                  "", _
                                  "Timeout=1000")
```

3.2.1.1.1. 接続時の注意点

LK-G3000LkIF プロバイダでは複数台のコントローラを追加した場合でも,通信先は1台のみとなり,複数の LK-G3000 と USB 接続した場合でもポート番号の小さい機器にのみ接続を行います. なおコントローラ追加時には通信は行いません.また Timeout オプションは KEYENCE 社製の通信ライブラリからの応答待ち時間となります.

3.2.2. CaoController クラス

3.2.2.1. VariableNames メソッド

接続可能な変数名リストを取得します. 本メソッドで取得した変数名は, 後述する AddVariable メソッドの第一引数に使用することができます.

使用例

```
' 変数名リスト取得
Dim variableNames() As String
variableNames = caoCtrl.variableNames
```

3.2.2.2. Variables プロパティ

コントローラが保持する, 変数コレクションを取得します.

使用例

```
' 変数コレクション取得
Dim variables As CaoVariables
Set variables = caoCtrl.variables
' 変数取得
```

```
Dim variable As CaoVariable
Set variable = variables.Item(0)
```

3.2.2.3. AddVariable メソッド

CaoController に変数オブジェクトを追加します. 変数名には 3.4.1 に示すもののみ使用できます. 以下に, AddVariable の仕様を示します.

書式

```
AddVariable
(
        "<変数名>",             // 変数名
        "<オプション>"          // オプション文字列(省略可能)
)
```

3.2.2.4. Execute メソッド

CaoController の拡張コマンドを実行します. また, LK-G3000LkIF プロバイダでは拡張コマンド実行時に機器と通信を行います. Execute で指定できる拡張コマンドについては 3.3. 拡張コマンド一覧に示すもののみ使用可能です. 以下に, Execute の仕様を示します.

書式

```
Execute
(
        "<拡張コマンド名>",             // 拡張コマンド名
        "<オプション文字列>"            // オプション文字列(省略可能)
)
```

3.2.3. CaoVariable クラス

3.2.3.1. Value プロパティ

LK-G3000 からデータを取得/設定します. また, LK-G3000LkIF プロバイダでは取得/設定時に機器と通信を行います. 変数名によって動作が異なります. 詳細は, 3.4. 変数一覧を参照してください.

3.3. 拡張コマンド一覧

使用可能な拡張コマンド一覧を定義します. 使用例は各コマンドの詳細で記述しています.

表 3-2 拡張コマンド一覧

コマンド	説明	参照
モード変更コマンド		

コマンド		説明	参照
	SetDisplayUnit	最小表示単位を設定します.	P.49
	GetDisplayUnit	最小表示単位を取得します.	P.50
	SetAnalogThrough	アナログスルーを設定します.	P.50
	GetAnalogThrough	アナログスルーを取得します.	P.51
共通設定関連コマンド			
	SetDataStorage	データストレージの対象 OUT, 蓄積点数, 蓄積周期を設定します.	P.51
	GetDataStorage	データストレージの対象 OUT, 蓄積点数, 蓄積周期を取得します.	P.52
	SetSamplingCycle	サンプリング周期を設定します.	P.53
	GetSamplingCycle	サンプリング周期を取得します.	P.53
	SetMutualInterPrev	相互干渉防止を設定します.	P.53
	GetMutualInterPrev	相互干渉防止を取得します.	P.54
	SetTimingSync	タイミング同期を設定します.	P.54
	GetTimingSync	タイミング同期を取得します.	P.55
	SetToleCompOutputFormat	判定出力形態を設定します.	P.55
	GetToleCompOutputFormat	判定出力形態を取得します.	P.55
	SetStorobeTime	ストローブ時間を設定します.	P.56
	GetStorobeTime	ストローブ時間を取得します.	P.56

3.3.1. モード変更コマンド

3.3.1.1. SetMode コマンド

本体の動作モードを変更します. ただし, 通信ライブラリ内では, 自動で動作モードの変更を行うためこのコマンドを明示的に実行する必要はありません. 以下に引数を示します.

項目	型説明	
引数	VT_I4	動作モードを指定します. 以下のいずれかを指定します. ・0 - 運転モード ・1 - 設定モード

使用例

```
// SetMode実行
Call caoCtrl.Execute("SetMode", 0)
```

3.3.2. 測定・制御関連コマンド

3.3.2.1. GetCalcData コマンド

測定値を取得します. 以下に戻り値を示します.

項目	型説明			
戻り値	VT_ARRAY | VT_VARIANT			
	0	VT_ARRAY | VT_VARIANT		OUT1 の測定値
		0	VT_I4	有効データかどうか. 以下のいずれかが取得されます. ・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
		1	VT_R4	測定値. 有効データ以外の場合は無効な値になります.
	1	VT_ARRAY | VT_VARIANT		OUT2 の測定値
		0	VT_I4	有効データかどうか. 上記の OUT1 の測定値と同様の値を取得します.
		1	VT_R4	測定値. 有効データ以外の場合は無効な値になります.

使用例

```
' GetCalcData実行
Dim values As Variant
values = caoCtrl.Execute("GetcalcData")

If Not IsEmpty(values) Then
        ' OUT1の測定値
        Dim value1 As Variant
        value1 = values(0)
        ' 有効データかどうか
        Dim validData1 As Long
        validData1 = value1(0)
        '測定値
        Dim fValue1 As Single
        fValue1 = value1(1)

        'OUT2の測定値
        Dim value2 As Variant
        value2 = values(1)
        '有効データかどうか
        Dim validData2 As Long
        validData2 = value2(0)
        ' 測定値
        Dim fValue2 As Single
        fValue2 = value2(1)
```

```
End If
```

3.3.2.2. SetTiming コマンド

タイミング信号の入力を設定します．タイミング入力を設定すると設定時の測定値を保持します．計測モードによってタイミング入力時の機能が異なります．タイミング入力の詳細は，LK-G3000 マニュアル内の「3 章 機能設定 － 測定値の出力条件を設定する － ホールド機能を使う(計測モード)」のタイミングチャートを参照してください．以下に引数を示します．

項目	型説明		
引数	VT_ARRAY | VT_VARIANT		
	0	VT_I4	設定する OUT 番号を指定します．以下のいずれかを指定してください． ・0 － OUT1 ・1 － OUT2 ・2 － OUT1 + OUT2
	1	VT_BOOL	タイミング入力を指定します．TRUE ならば ON，FALSE なら OFF を指定します．

使用例

```
' SetTiming実行
Dim param As Variant
param = Array(0, True)
Call caoCtrl.Execute("SetTiming", param)
```

3.3.2.3. SetZero コマンド

オートゼロを設定します．オートゼロを ON にすると測定中の測定値をゼロにします．以下に引数を示します．

項目	型説明		
引数	VT_ARRAY | VT_VARIANT		
	0	VT_I4	設定する OUT 番号を指定します．以下のいずれかを指定してください． ・0 － OUT1 ・1 － OUT2 ・2 － OUT1 + OUT2
	1	VT_BOOL	オートゼロを指定します．TRUE ならば ON，FALSE なら OFF を指定します．

使用例

```
' SetZero実行
Dim param As Variant
```

```
param = Array(0, True)
Call caoCtrl.Execute("SetZero", param)
```

3.3.2.4. SetReset コマンド

リセット入力を設定し，指定した OUT 番号の測定値をリセットします．以下に引数を示します．

項目	型説明	
引数	VT_I4	設定する OUT 番号を指定します．以下のいずれかを指定してください． ・0 - OUT1 ・1 - OUT2 ・2 - OUT1 + OUT2

使用例

```
' SetReset実行
Call caoCtrl.Execute("SetReset", 0)
```

3.3.2.5. SetPanelLock コマンド

表示パネルのキー操作をロックします．表示パネルをロックすることで，誤って操作キーに触れても誤操作を防止できます．以下に引数を示します．

項目	型説明	
引数	VT_BOOL	パネルロックを指定します．TRUE ならば ON，FALSE なら OFF を指定します．

使用例

```
' SetPanelLock実行
Call caoCtrl.Execute("SetPanelLock", true)
```

3.3.2.6. SetProgramNo コマンド

プログラム番号を切り替えます．以下に引数を示します．

項目	型説明	
引数	VT_I4	切り替えたいプログラム番号を指定します．0～7 までの値を指定できます．

使用例

```
' SetProgramNo実行
Call caoCtrl.Execute("SetProgramNo", 0)
```

3.3.2.7. GetProgramNo コマンド

現在のプログラム番号を取得します．以下に引数を示します．

項目	型説明	
戻り値	VT_I4	プログラム番号を取得します．

使用例

```
' GetProgramNo実行
Dim value As Integer
value = caoCtrl.Execute("GetProgramNo")
```

3.3.2.8. GetFigureData コマンド

統計処理をした測定値の統計結果を取得します．統計処理の対象となるデータは，各測定モードにおいてホールドされたデータです．以下に引数と戻り値を示します．

項目	型説明			
引数	VT_I4			OUT 番号を指定します．以下のいずれかを指定してください． ・0 - OUT1 ・1 - OUT2
戻り値	VT_ARRAY | VT_VARIANT			
	0	VT_ARRAY | VT_VARIANT		公差上限値
		0	VT_I4	有効データかどうか．以下のいずれかが取得されます． ・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
		1	VT_R4	数値．有効データ以外の場合は無効な値になります．
	1	VT_ARRAY | VT_VARIANT		公差下限値
		0	VT_I4	有効データかどうか．上記の公差上限値と同様の値を取得します．
		1	VT_R4	数値．有効データ以外の場合は無効な値になります．
	2	VT_ARRAY | VT_VARIANT		平均値
		0	VT_I4	有効データかどうか．上記の公差上限値と同様の値を取得します．
		1	VT_R4	数値．有効データ以外の場合は無効な値になります．
	3	VT_ARRAY | VT_VARIANT		最大値
		0	VT_I4	有効データかどうか．上記の公差上限値と同様の値を取得します．

項目	型説明			
		1	VT_R4	数値．有効データ以外の場合は無効な値になります．
	4	VT_ARRAY | VT_VARIANT		最小値
		0	VT_I4	有効データかどうか．以下のいずれかが取得されます． ・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
		1	VT_R4	数値．有効データ以外の場合は無効な値になります．
	5	VT_ARRAY | VT_VARIANT		最大値 － 最小値
		0	VT_I4	有効データかどうか．上記の公差上限値と同様の値を取得します．
		1	VT_R4	数値．有効データ以外の場合は無効な値になります．
	6	VT_ARRAY | VT_VARIANT		標準偏差
		0	VT_I4	有効データかどうか．上記の公差上限値と同様の値を取得します．
		1	VT_R4	数値．有効データ以外の場合は無効な値になります．
	7	VT_I4		総データ数
	8	VT_I4		High 判定データ数
	9	VT_I4		Go 判定データ数
	10	VT_I4		Low 判定データ数

使用例

```
' GetFigureData実行
Dim values As Variant
values = caoCtrl.Execute("GetFigureData", 0)

If Not IsEmpty(values) Then
        ' 公差上限値
        Dim maxLimit As Variant
        maxLimit = values(0)
        ' 有効データかどうか
        Dim validData1 As Long
        validData1 = maxLimit(0)
        ' 測定値
        Dim fValue1 As Single
        fValue1 = maxLimit(1)

        ' 公差下限値
        Dim minLimit As Variant
```

```
minLimit = values(1)
' 有効データかどうか
Dim validData2 As Long
validData2 = minLimit(0)
' 測定値
Dim fValue2 As Single
fValue2 = minLimit(1)

' 平均値
Dim average As Variant
average = values(2)
' 有効データかどうか
Dim validData3 As Long
validData3 = average(0)
' 測定値
Dim fValue3 As Single
fValue3 = average(1)

' 最大値
Dim maxValue As Variant
maxValue = values(3)
' 有効データかどうか
Dim validData4 As Long
validData4 = maxValue(0)
' 測定値
Dim fValue4 As Single
fValue4 = maxValue(1)

' 最小値
Dim minValue As Variant
minValue = values(4)
' 有効データかどうか
Dim validData5 As Long
validData5 = minValue(0)
' 測定値
Dim fValue5 As Single
fValue5 = minValue(1)

' 最大値 - 最小値
Dim difValue As Variant
difValue = values(5)
' 有効データかどうか
Dim validData6 As Long
validData6 = difValue(0)
' 測定値
Dim fValue6 As Single
fValue6 = difValue(1)
```

```
        ' 標準偏差
        Dim stndDev As Variant
        stndDev = values(6)
        ' 有効データかどうか
        Dim validData7 As Long
        validData7 = stndDev(0)
        ' 測定値
        Dim fValue7 As Single
        fValue7 = stndDev(1)

        ' 総データ数
        Dim totalDataCnt As Integer
        totalDataCnt = values(7)
        ' High判定データ数
        Dim highDataCnt As Integer
        highDataCnt = values(8)
        ' Go判定データ数
        Dim goDataCnt As Integer
        goDataCnt = values(9)
        ' Low判定データ数
        Dim lowDataCnt As Integer
        lowDataCnt = values(10)
End If
```

3.3.2.9. ClearFigureData コマンド

統計値をクリアします.

使用例

```
' ClearFigureData実行
Call caoCtrl.Execute("ClearFigureData")
```

3.3.2.10. StartDataStorage コマンド

データストレージにデータの蓄積を開始します.

使用例

```
' StartDataStorage実行
Call caoCtrl.Execute("StartDataStorage")
```

3.3.2.11. StopDataStorage コマンド

データストレージにデータの蓄積を停止します.

使用例

```
' StopDataStorage実行
Call caoCtrl.Execute("StopDataStorage")
```

3.3.2.12. ClearDataStorage 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.

使用例

```
' ClearDataStorage実行
Call caoCtrl.Execute("ClearDataStorage")
```

3.3.2.13. GetDataStorageData コマンド

データストレージの蓄積データを取得します. 以下に引数と戻り値を示します.

項目	型説明			
引数	VT_ARRAY | VT_I4			
	0	VT_I4		OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2
	1	VT_I4		取得するデータの個数を指定します. 1～65536 まで指定可能です.
戻り値	VT_ARRAY | VT_VARIANT			
	0	VT_I4		取得した蓄積データの数
	i	VT_ARRAY | VT_VARIANT		i 個目の蓄積データ
		0	VT_I4	有効データかどうか. 以下のいずれかが取得されます. ・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
		1	VT_R4	数値. 有効データ以外の場合は無効な値になります.

※ i - 取得した蓄積データ数分

使用例

```
' GetDataStorageData実行
Dim param As Variant
param = Array(0, 3)
Dim value As Variant
value = caoCtrl.Execute("GetDataStorageData", param)
If Not IsEmpty(value) Then
        ' 取得した蓄積データ数
        Dim readDataCnt As Integer
        readDataCnt = value(LBound (value))
        Dim i As Integer
        For i = LBound(value) + 1 To (UBound(value))
                ' 蓄積データ
                Dim accumulationData As Variant
                accumulationData = value(i)
```

```
                If Not IsEmpty(accumulationData) Then
                        ' 有効データかどうか
                        Dim result As Integer
                        result = accumulationData(0)
                        ' 測定値
                        Dim fValue As Single
                        fValue = accumulationData(1)
                End If
        Next i
End If
```

3.3.2.14. GetDataStorageStatus コマンド

データストレージの蓄積状態を取得します. 以下に引数と戻り値を示します.

項目	型説明		
引数	VT_I4		OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2
戻り値	VT_ARRAY | VT_VARIANT		
	0	VT_BOOL	蓄積中かどうかを取得します. ・TRUE - 蓄積中 ・FALSE - 停止中
	1	VT_I4	蓄積されているデータ件数

使用例

```
' GetDataStorageStatus実行
Dim value As Variant
value = caoCtrl.Execute("GetDataStorageStatus", 0)
If Not IsEmpty(value) Then
        ' 蓄積中かどうか
        Dim isAccumulation As Boolean
        isAccumulation = value(LBound(value))
        ' 蓄積されているデータ件数
        Dim dataCnt As Integer
        dataCnt = value(UBound(value))
End If
```

3.3.2.15. GetLight コマンド

受光波形を取得します. 以下に引数と戻り値を示します.

項目	型説明
引数	VT_ARRAY | VT_I4

項目	型説明			
	0	VT_I4		ヘッド番号を指定します．以下のいずれかを指定してください． ・0 - HEAD-A ・1 - HEAD-B
	1	VT_I4		取得するデータの個数を指定します．1～65536 まで指定可能です．
戻り値	VT_ARRAY | VT_VARIANT			
	0	VT_ARRAY | VT_I4		測定位置
		0	VT_I4	ピーク番号 0 の測定位置
		1	VT_I4	ピーク番号 1 の測定位置
		2	VT_I4	ピーク番号 2 の測定位置
		3	VT_I4	ピーク番号 3 の測定位置
	1	VT_I4		有効データ数
	2	VT_ARRAY | VT_UI1		読み出したデータ
		0	VT_I4	1 番目のデータ
		…	…	…
		1023	VT_I4	1024 番目のデータ

```
' GetLight実行
Dim param As Variant
param = Array(0, 0)
Dim value As Variant
value = caoCtrl.Execute("GetLight", param)
If Not IsEmpty(value) Then
        ' 測定位置
        Dim mesurePosition As Variant
        mesurePosition = value(0)
        If Not IsEmpty(value) Then
                ' ピーク番号0の測定位置
                Dim peekPosition0 As Integer
                peekPosition0 = mesurePosition(0)
                ' ピーク番号1の測定位置
                Dim peekPosition1 As Integer
                peekPosition1 = mesurePosition(1)
                ' ピーク番号2の測定位置
                Dim peekPosition2 As Integer
                peekPosition2 = mesurePosition(2)
                ' ピーク番号3の測定位置
                Dim peekPosition3 As Integer
                peekPosition3 = mesurePosition(3)
        End If

        ' 有効データ数
```

```
        Dim effDataCnt As Integer
        effDataCnt = value(1)

        ' 読み出したデータ
        Dim allReadData As Variant
        allReadData = value(2)
        If Not IsEmpty(value) Then
                Dim i As Integer
                For i = 0 To 1023
                        Dim readData As Byte
                        readData = allReadData(i)
                Next i
        End If
End If
```

3.3.3. 設定内容関連コマンド

3.3.3.1. パネル表示関連コマンド

3.3.3.1.1. SetPanel コマンド

パネル表示を切り替えます. 以下に引数を示します.

項目	型説明	
引数	VT_I4	表示させる OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2 ・2 - OUT1+OUT2

使用例

```
' SetPanel実行
Call caoCtrl.Execute("SetPanel", 0)
```

3.3.3.1.2. GetPanel コマンド

表示中のパネルを取得します. 以下に戻り値を示します.

項目	型説明	
戻り値	VT_I4	表示中の OUT 番号を取得します. 値の詳細は 3.3.3.1.1 を参照してください.

使用例

```
' GetPanel実行
Dim outNo As Integer
outNo = caoCtrl.Execute("GetPanel")
```

3.3.3.2. 公差設定関連コマンド

3.3.3.2.1. SetTolerance コマンド

許容範囲の判定値(公差判定値)を設定します．それぞれ上限値を超えたとき(HI)，下限値を超えたとき(LO)，許容範囲(GO)の 3 段階に判定し，表示と出力をすることができます．また，測定値が公差判定値の付近で上下している場合は，判定出力が ON/OFF を繰り返すことがあります．ヒステリシスを設定すると公差判定の検出値と復帰値に幅ができるので，このような状態を防ぐことができます．表示内容及びヒステリシスに関しましては，LK-G3000 マニュアル内の「2 章 － 公差判定値を設定する」を参照してください．以下に引数を示します．

項目	型説明		
引数	VT_ARRAY | VT_I4		
	0	VT_I4	OUT 番号を指定します．以下のいずれかを指定してください． ・0 － OUT1 ・1 － OUT2
	1	VT_I4	公差上限値を指定します． -999999～999999 の値を指定します．
	2	VT_I4	公差下限値を指定します． -999999～999999 の値を指定します．
	3	VT_I4	公差ヒステリシスを指定します． 0～999999 の値を指定します．

※ 「公差上限値 －（公差下限値）＞ 公差ヒステリシス」となるように指定しなければ API エラーが返されます．以下に例を記述します．

良い例 -> 公差上限値 = 1000, 公差下限値 = 100, 公差ヒステリシス = 0

悪い例 -> 公差上限値 = -1000, 公差下限値 = 100, 公差ヒステリシス = 0

使用例

```
' SetTolerance実行
Dim param As Variant
param = Array(0, 1000, 100, 0)
Call caoCtrl.Execute("SetTolerance", param)
```

3.3.3.2.2. GetTolerance コマンド

許容範囲の判定値(公差判定値)を取得します．以下に引数と戻り値を示します．

項目	型説明

引数	VT_I4		表示させる OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2
戻り値	VT_ARRAY | VT_I4		
	0	VT_I4	公差上限値を取得します.
	1	VT_I4	公差下限値を取得します.
	2	VT_I4	公差ヒステリシスを取得します.

使用例

```
' GetTolerance実行
Dim tolerance As Variant
tolerance = caoCtrl.Execute("GetTolerance", 0)
If Not IsEmpty(tolerance) Then
        ' 公差上限値
        Dim maxLimit As Integer
        maxLimit = tolerance(0)
        ' 公差下限値
        Dim minLimit As Integer
        minLimit = tolerance(1)
        ' 公差ヒステリシス
        Dim hysterisis As Integer
        hysterisis = tolerance(2)
End If
```

3.3.3.3. ヘッド設定関連コマンド

安定した検出を行うためのセンシングに関連する機能を設定/取得するコマンドです.

3.3.3.3.1. SetAbleMode コマンド

ABLE チューニングモードを設定します. ABLE 機能は, 対象物の表面状態(色, 光沢, 材質)に適切な光量と感度に自動調整します. 以下に引数を示します.

項目	型説明		
引数	VT_ARRAY | VT_I4		
	0	VT_I4	ヘッド番号を指定します. 以下のいずれかを指定してください. ・0 - HEAD-A ・1 - HEAD-B

項目	型説明		
	1	VT_I4	ABLE チューニングモードを指定します. ABLE チューニングモードについては, 表 3-3 を参照してください. 以下のいずれかを指定してください. ・0 - 自動 ・1 - マニュアル

表 3-3 ABLE チューニングモード詳細

モード	機能
自動	自動で適切な光量に調節します. 通常はこちらを選択します.
マニュアル	光量と感度の調整範囲を 1 ～ 99 の任意の範囲に限定して調整します. 対象物の反射率が早い周期で大きく変化する場合や, 対象物のみを検出する場合に選択します.

使用例

```
' SetAbleMode実行
Dim param As Variant
param = Array(0, 0)
Call caoCtrl.Execute("SetAbleMode", param)
```

3.3.3.3.2. GetAbleMode コマンド

ABLE チューニングモードを取得します. 以下に引数と戻り値を示します.

項目	型説明	
引数	VT_I4	ヘッド番号を指定します. 以下のいずれかを指定してください. ・0 - HEAD-A ・1 - HEAD-B
戻り値	VT_I4	ABLE チューニングモードを取得します. 値の詳細は 3.3.3.3.1 を参照してください.

使用例

```
' GetAbleMode実行
Dim mode As Integer
mode = caoCtrl.Execute("GetAbleMode", 0)
```

3.3.3.3.3. SetAbleMinMax コマンド

ABLE 制御範囲を設定します. 以下に引数を示します.

項目	型説明

項目	型説明		
引数	VT_ARRAY | VT_I4		
	0	VT_I4	ヘッド番号を指定します. 以下のいずれかを指定してください. ・0 - HEAD-A ・1 - HEAD-B
	1	VT_I4	ABLE 最小値を指定します. 1～99 の値を指定します.
	2	VT_I4	ABLE 最大値を指定します. 1～99 の値を指定します.

※ 「ABLE 最大値 - ABLE 最小値 >= 0」以外は API エラーが返ってきます.

使用例

```
' SetAbleMinMax実行
Dim param As Variant
param = Array(0, 8, 10)
Call caoCtrl.Execute("SetAbleMinMax", param)
```

3.3.3.3.4. GetAbleMinMax コマンド

ABLE 制御範囲を取得します. 以下に引数と戻り値を示します.

項目	型説明		
引数	VT_I4		ヘッド番号を指定します. 以下のいずれかを指定してください. ・0 - HEAD-A ・1 - HEAD-B
戻り値	VT_ARRAY | VT_I4		
	0	VT_I4	ABLE 最小値を取得します.
	1	VT_I4	ABLE 最大値を取得します.

使用例

```
' GetAbleMinMax実行
Dim value As Variant
value = caoCtrl.Execute("GetAbleMinMax", 0)
If Not IsEmpty(value) Then
        ' ABLE最小値
        Dim minAble As Integer
        minAble = value(0)
        ' ABLE最大値
        Dim maxAble As Integer
        maxAble = value(1)
End If
```

3.3.3.3.5. SetMeasureMode コマンド

測定対象物に合わせて測定モードを設定します．測定する対象物がどのようなものかを指定することで，安定した検出を行います．以下に引数を示します．

項目	型説明		
引数	VT_ARRAY | VT_I4		
	0	VT_I4	ヘッド番号を指定します．以下のいずれかを指定してください． ・0 - HEAD-A ・1 - HEAD-B
	1	VT_I4	測定モードを指定します．各測定モードの詳細については表 3-4 を参照してください．以下のいずれかを指定してください． ・0 - 標準 ・1 - 半透明体 ・2 - 透明体 ・3 - 透明体 2 ・4 - 多重反射体

表 3-4 測定モード詳細

モード	機能
標準	通常はこの設定を使用します．
半透明体	半透明樹脂など，光を吸収するような対象物に対応します．
透明体	透明体の変異測定や厚み測定に使用します． 透明体の複数面の反射率が同党の場合に使用します．
透明体 2	透明体の表面や裏面などの複数面(最大 4 面)の反射率が異なる場合などに使用します．
多重反射体	IC やコネクタの端子の曲がり測定などに使用します．

使用例

```
' SetMeasureMode実行
Dim param As Variant
param = Array(0, 0)
Call caoCtrl.Execute("SetMeasureMode", param)
```

3.3.3.3.6. GetMeasureMode コマンド

測定モードを取得します．以下に引数と戻り値を示します．

項目	型説明

項目	型説明	
引数	VT_I4	ヘッド番号を指定します．以下のいずれかを指定してください． ・0 - HEAD-A ・1 - HEAD-B
戻り値	VT_I4	測定モードを取得します．値の詳細は 3.3.3.3.5 を参照してください．

使用例

```
' GetMeasureMode実行
Dim mode As Integer
mode = caoCtrl.Execute("GetMeasureMode", 0)
```

3.3.3.3.7. SetNumAlarm コマンド

アラーム処理回数を設定します．以下に引数を示します．

項目	型説明		
引数	VT_ARRAY | VT_I4		
	0	VT_I4	ヘッド番号を指定します．以下のいずれかを指定してください． ・0 - HEAD-A ・1 - HEAD-B
	1	VT_I4	アラーム処理回数を指定します． 0～999 の値を指定します．

使用例

```
' SetNumAlarm実行
Dim param As Variant
param = Array(0, 5)
Call caoCtrl.Execute("SetNumAlarm", param)
```

3.3.3.3.8. GetNumAlarm コマンド

アラーム処理回数を取得します．以下に引数と戻り値を示します．

項目	型説明	
引数	VT_I4	ヘッド番号を指定します．以下のいずれかを指定してください． ・0 - HEAD-A ・1 - HEAD-B
戻り値	VT_I4	アラーム処理回数を取得します．

使用例

```
' GetNumAlarm実行
Dim numAlram As Integer
numAlram = caoCtrl.Execute("GetNumAlarm", 0)
```

3.3.3.3.9. SetAlarmLevel コマンド

アラームレベルを設定します. 以下に引数を示します.

項目	型説明		
引数	VT_ARRAY | VT_I4		
	0	VT_I4	ヘッド番号を指定します. 以下のいずれかを指定してください. ・0 - HEAD-A ・1 - HEAD-B
	1	VT_I4	アラームレベルを指定します. 0～9 の値を指定します. 大きな値になるほどアラームになりやすくなります.

使用例

```
' SetAlarmLevel実行
Dim param As Variant
param = Array(0, 3)
Call caoCtrl.Execute("SetAlarmLevel", param)
```

3.3.3.3.10. GetAlarmLevel コマンド

アラームレベルを取得します. 以下に引数と戻り値を示します.

項目	型説明	
引数	VT_I4	ヘッド番号を指定します. 以下のいずれかを指定してください. ・0 - HEAD-A ・1 - HEAD-B
戻り値	VT_I4	アラームレベルを取得します.

使用例

```
' GetAlarmLevel実行
Dim level As Integer
level = caoCtrl.Execute("GetAlarmLevel", 0)
```

3.3.3.3.11. StartABLE コマンド

ABLE チューニングを開始します. ABLE チューニング機能は, 対象のヘッドが実際に対象物を測定することで, ABLE の調整範囲を最適化します. 図 3-1 を参考に ABLE チューニングを行ってください. ABLE チューニングを実行すると, ABLE チューニングモードはマニュアルに, ABLE 上限値, ABLE 下限

値は調整された値に設定されます.このコマンドは,本体の状態を通信モードにしたまま終了します.このコマンドを実行した場合は,必ずStopABLEコマンドもしくは,CancelABLEコマンドを実行して,本体を通常モードにしてください. 以下に引数を示します.

項目	型説明	
引数	VT_I4	ヘッド番号を指定します. 以下のいずれかを指定してください. ・0 - HEAD-A ・1 - HEAD-B

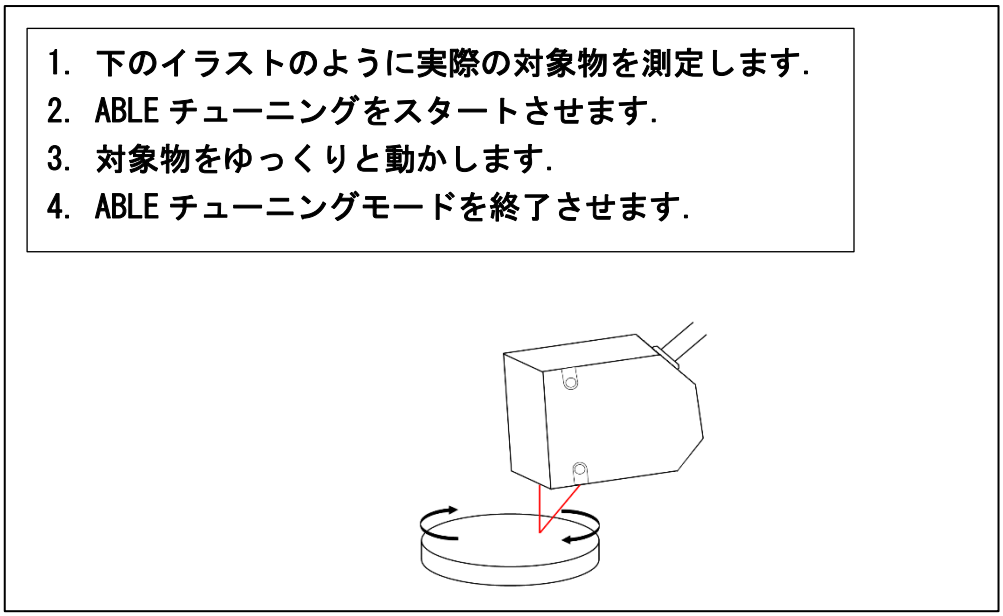


図 3-1 ABLE チューニングの流れ

使用例

```
' StartABLE実行
Call caoCtrl.Execute("StartABLE", 0)
```

3.3.3.3.12. StopABLE コマンド

ABLE チューニングを終了します.

使用例

```
' StopABLE実行
Call caoCtrl.Execute("StopABLE", "")
```

3.3.3.3.13. CancelABLE コマンド

ABLE チューニングを中止します.

使用例

```
' CancelABLE実行
```

```
Call caoCtrl.Execute("CancelABLE", "")
```

3.3.3.3.14. SetReflectionMode コマンド

設置モードを設定します. 以下に引数を示します.

項目	型説明		
引数	VT_ARRAY | VT_I4		
	0	VT_I4	ヘッド番号を指定します. 以下のいずれかを指定してください. ・0 - HEAD-A ・1 - HEAD-B
	1	VT_I4	設置モードを指定します. 設置モードの詳細については表 3-5 を参照してください. 以下のいずれかを指定してください. ・0 - 拡散反射 ・1 - 正反射

表 3-5 設置モード詳細

モード	機能
拡散反射	拡散反射を設定します. 通常はこの設定を選択します.
正反射	正反射を設定します. 測定対象物が鏡面やガラスなどのときに選択します.

使用例

```
' SetReflectionMode実行
Dim param As Variant
param = Array(0, 0)
Call caoCtrl.Execute("SetReflectionMode", param)
```

3.3.3.3.15. GetReflectionMode コマンド

設置モードを取得します. 以下に引数と戻り値を示します.

項目	型説明	
引数	VT_I4	ヘッド番号を指定します. 以下のいずれかを指定してください. ・0 - HEAD-A ・1 - HEAD-B
戻り値	VT_I4	設置モードを取得します. 値の詳細は 3.3.3.3.14 を参照してください.

使用例

```
' GetReflectionMode実行
Dim mode As Integer
```

```
mode = caoCtrl.Execute("GetReflectionMode", 0)
```

3.3.3.4. OUT 設定関連コマンド

データ処理に関連する機能を設定/取得するコマンドです.

3.3.3.4.1. SetCalcMethod コマンド

ヘッド間の演算方法を設定します. 測定する対象物に応じて, ヘッド A, またはヘッド B のヘッド設定で得られたデータを演算することで表面変位, 厚み, 段差測定ができます. 以下に引数を示します.

項目	型説明		
引数	VT_ARRAY | VT_I4		
	0	VT_I4	OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2
	1	VT_I4	演算方法を指定します. 演算方法の詳細については表 3-6 を参照してください. 以下のいずれかを指定してください. ・0 - ヘッド A ・1 - ヘッド B ・2 - ヘッド A+ヘッド B ・3 - ヘッド A-ヘッド B ・4 - ヘッド A 透明体 ・5 - ヘッド B 透明体
	2	VT_I4	測定対象を指定します. 測定対象の詳細については表 3-7 を参照してください. 以下のいずれかを指定してください. ・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

表 3-6 演算方法詳細

演算	機能

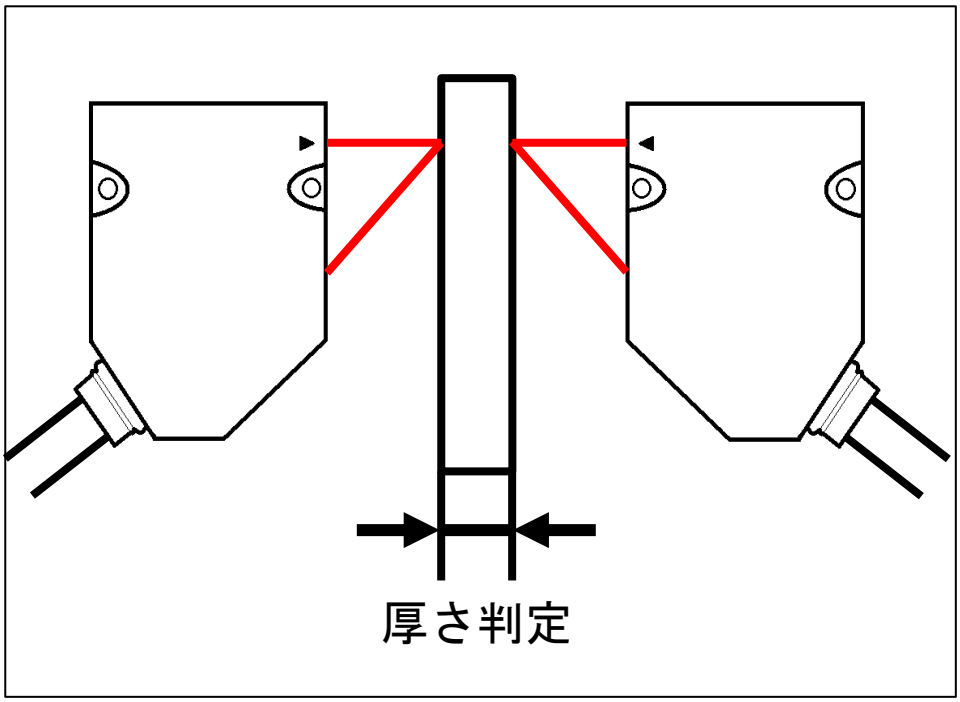
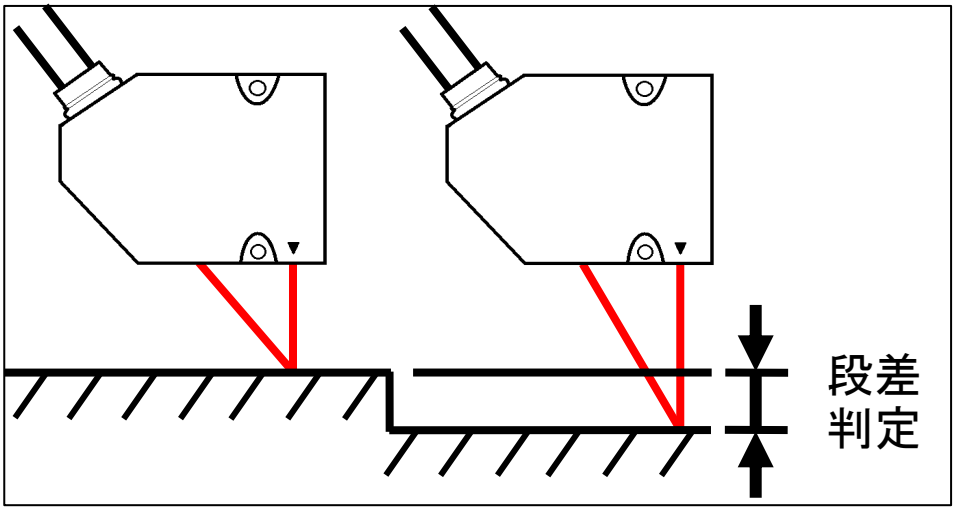
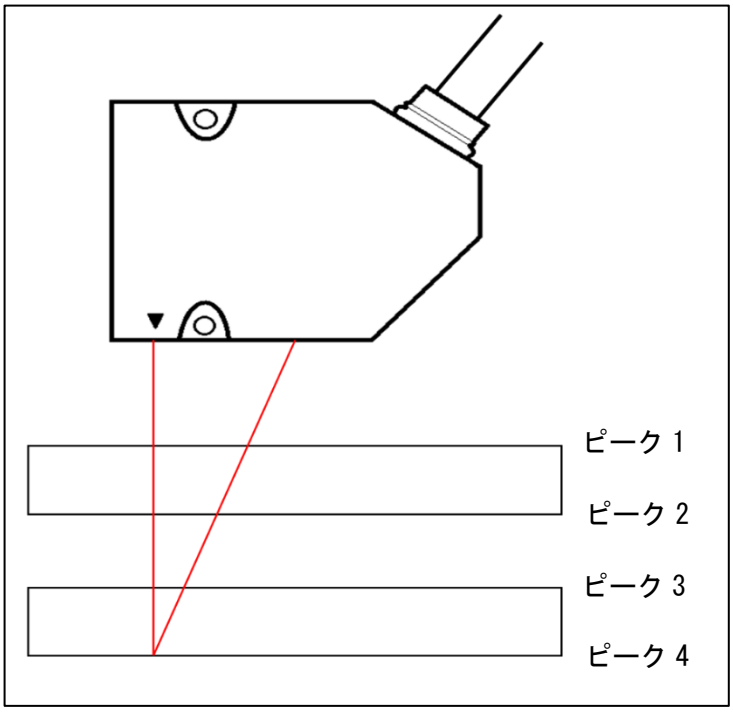
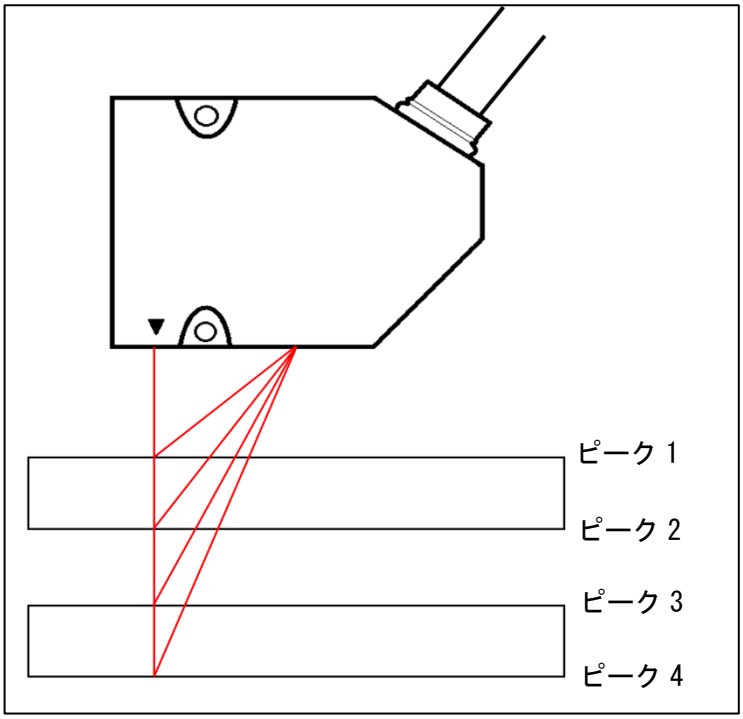
演算	機能
ヘッド A	ヘッド A，またはヘッド B の表面変位測定
ヘッド B	
ヘッド A+ヘッド B	ヘッド A とヘッド B を使用した厚さ判定 
ヘッド A-ヘッド B	ヘッド A とヘッド B を使用した厚さ判定 
ヘッド A 透明体	透明体の変位測定や厚さ測定をします．対象面の選択は，測定対象にて指定します．
ヘッド B 透明体	

表 3-7 測定対象詳細

測定面	機能

<table>
<tr><th>測定面</th><th>機能</th></tr>
<tr><td>ピーク 1
ピーク 2
ピーク 3
ピーク 4</td><td>1 反射面の変位測定
 
</td></tr>
<tr><td>ピーク 1 － 2
ピーク 1 － 3
ピーク 1 － 4
ピーク 2 － 3
ピーク 2 － 4
ピーク 3 － 4</td><td>2 反射面の測定と「ピーク 1 － 2」を選択すれば，1 枚目のガラスの厚みを測定できます．また，「ピーク 2 － 3」を選択すれば，1 枚目と 2 枚目のガラスの間隔を測定できます．
 
</td></tr>
</table>

使用例

```
' SetCalcMethod実行
Dim param As Variant
param = Array(0, 0, 0)
Call caoCtrl.Execute("SetCalcMethod", param)
```

3.3.3.4.2. GetCalcMethod コマンド

演算方法を取得します．以下に引数と戻り値を示します．

項目	型説明		
引数	VT_I4		OUT 番号を指定します．以下のいずれかを指定してください． ・0 - OUT1 ・1 - OUT2
戻り値	VT_ARRAY | VT_I4		
	0	VT_I4	演算方法を取得します．値の詳細は 3.3.3.4.1 を参照してください．
	1	VT_I4	測定対象を取得します．値の詳細は 3.3.3.4.1 を参照してください．

使用例

```
' GetCalcMethod実行
Dim value As Variant
value = caoCtrl.Execute("GetCalcMethod", 0)
If Not IsEmpty(value) Then
        ' 演算方法
        Dim method As Integer
        method = value(0)
        ' 測定対象
        Dim target As Integer
        target = value(1)
End If
```

3.3.3.4.3. SetScaling コマンド

スケーリングを設定します．スケーリングを設定することで，測定値に対する表示値を任意に校正することができます．校正は任意の 2 点のポイントに対してそれぞれ表示させる値を設定します．OUT1，OUT2 それぞれに対してヘッド A，ヘッド B の校正が出来ます．スケーリングの詳細につきましては，LK-G3000 マニュアルを参照してください．以下に引数を示します．

項目	型説明		
引数	VT_ARRAY | VT_I4		
	0	VT_I4	OUT 番号を指定します．以下のいずれかを指定してください． ・0 - OUT1 ・1 - OUT2

項目	型説明		
	1	VT_I4	ヘッド番号を指定します．以下のいずれかを指定してください． ・0 - HEAD-A ・1 - HEAD-B
	2	VT_I4	入力値 1 を指定します． -999999～999999 の値を指定します．
	3	VT_I4	表示値 1 を指定します． -999999～999999 の値を指定します．
	4	VT_I4	入力値 2 を指定します． -999999～999999 の値を指定します．
	5	VT_I4	表示値 2 を指定します． -999999～999999 の値を指定します．

※ 以下の条件を満たさない場合，設定に失敗し，API エラーが返ってきます．

(1) 入力値 1 - 入力値 2 ≠ 0

使用例

```
' SetScaling実行
Dim param As Variant
param = Array(0, 0, 500, 10, 1000, 1000)
Call caoCtrl.Execute("SetScaling", param)
```

3.3.3.4.4. GetScaling コマンド

スケーリングを取得します．以下に引数と戻り値を示します．

項目	型説明		
引数	VT_ARRAY | VT_I4		
	0	VT_I4	OUT 番号を指定します．以下のいずれかを指定してください． ・0 - OUT1 ・1 - OUT2
	1	VT_I4	ヘッド番号を指定します．以下のいずれかを指定してください． ・0 - HEAD-A ・1 - HEAD-B
戻り値	VT_ARRAY | VT_I4		
	0	VT_I4	入力値 1 を取得します．
	1	VT_I4	表示値 1 を取得します．
	2	VT_I4	入力値 2 を取得します．
	3	VT_I4	表示値 2 を取得します．

使用例

```
' GetScaling実行
Dim param As Variant
param = Array(0, 0)
Dim value As Variant
value = caoCtrl.Execute("GetScaling", param)
If Not IsEmpty(value) Then
        ' 入力値1
        Dim inputValue1 As Integer
        inputValue1 = value(0)
        ' 表示値1
        Dim outputValue1 As Integer
        outputValue1 = value(1)
        ' 入力値2
        Dim inputValue2 As Integer
        inputValue2 = value(2)
        ' 表示値2
        Dim outputValue2 As Integer
        outputValue2 = value(3)
End If
```

3.3.3.4.5. SetFilterMode コマンド

フィルタモードを設定します.フィルタをかけることにより,安定した測定を行うことができます.フィルタ機能の詳細に関しては,LK-G3000 マニュアル内の「3 章 -測定値の出力条件を設定する - フィルタをかけて安定した測定をする」を参照してください. 以下に引数を示します.

項目	型説明		
引数	VT_ARRAY | VT_I4		
	0	VT_I4	OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2
	1	VT_I4	フィルタモードを指定します. 以下のいずれかを指定してください. ・0 - 移動平均 ・1 - ローパスフィルタ ・2 - ハイパスフィルタ

使用例

```
' SetFilterMode実行
Dim param As Variant
param = Array(0, 0)
Call caoCtrl.Execute("SetFilterMode", param)
```

3.3.3.4.6. GetFilterMode コマンド

フィルタモードを取得します. 以下に引数と戻り値を示します.

項目	型説明	
引数	VT_I4	OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2
戻り値	VT_I4	フィルタモードを取得します. 値の詳細は SetFilterMode コマンドを参照してください.

使用例

```
' GetFilterMode実行
Dim mode As Integer
mode = caoCtrl.Execute("GetFilterMode", 0)
```

3.3.3.4.7. SetAverage コマンド

フィルタモードが移動平均の場合の平均回数を設定します. 以下に引数を示します.

項目	型説明		
引数	VT_ARRAY | VT_I4		
	0	VT_I4	OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2
	1	VT_I4	平均回数を指定します. 以下のいずれかを指定してください. ・0 - 1 回 ・1 - 4 回 ・2 - 16 回 ・3 - 64 回 ・4 - 256 回 ・5 - 1024 回 ・6 - 4096 回 ・7 - 16384 回 ・8 - 65536 回 ・9 - 262144 回

使用例

```
' SetAverage実行
Dim param As Variant
param = Array(0, 0)
```

```
Call caoCtrl.Execute("SetAverage", param)
```

3.3.3.4.8. GetAverage コマンド

フィルタモードが移動平均の平均回数を取得します. 以下に引数と戻り値を示します.

項目	型説明	
引数	VT_I4	OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2
戻り値	VT_I4	平均回数を取得します. 値の詳細は 3.3.3.4.7 を参照してください.

使用例

```
' GetAverage実行
Dim average As Integer
average = caoCtrl.Execute("GetAverage", 0)
```

3.3.3.4.9. SetCutOffFrequency コマンド

フィルタモードがローパスフィルター/ハイパスフィルター設定時のカットオフ周波数を設定します. 以下に引数を示します.

項目	型説明		
引数	VT_ARRAY | VT_I4		
	0	VT_I4	OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2
	1	VT_I4	カットオフ周波数を指定します. 以下のいずれかを指定してください. ・0 - 1000Hz ・1 - 300Hz ・2 - 100Hz ・3 - 30Hz ・4 - 10Hz ・5 - 3Hz ・6 - 1Hz ・7 - 0.3Hz ・8 - 0.1Hz

使用例

```
' SetCutOffFrequency実行
```

```
Dim param As Variant
param = Array(0, 0)
Call caoCtrl.Execute("SetCutOffFrequency", param)
```

3.3.3.4.10. GetCutOffFrequency コマンド

フィルタモードがローパスフィルター/ハイパスフィルター設定時のカットオフ周波数を取得します. 以下に引数と戻り値を示します.

項目	型説明	
引数	VT_I4	OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2
戻り値	VT_I4	カットオフ周波数を取得します. 値の詳細は 3.3.3.4.9 を参照してください.

使用例

```
' GetCutOffFrequency実行
Dim frequency As Integer
frequency = caoCtrl.Execute("GetCutOffFrequency", 0)
```

3.3.3.4.11. SetTriggerMode コマンド

トリガモードを設定します. 以下に引数を示します.

項目	型説明		
引数	VT_ARRAY | VT_I4		
	0	VT_I4	OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2
	1	VT_I4	トリガモードを指定します. トリガモードの詳細については, 表 3-8 を参照してください以下のいずれかを指定してください. ・0 - 外部トリガ 1 ・1 - 外部トリガ 2

表 3-8 トリガモード詳細

モード	機能		
	標準	ピークホールド/ボトムホールド/ピーク to ピークホールド/アベレージホールド	サンプルホールド
トリガ 1	タイミング入力が立ち	タイミング入力が立ち上がり	タイミング入力が立ち上

モード	機能		
	標準	ピークホールド/ボトムホールド/ピーク to ピークホールド/アベレージホールド	サンプルホールド
	上がった(ON)ときの内部測定値を ON になっている期間ホールドします.	エッジから次の立ち上がりエッジまでをサンプリング期間とします.	がったときの内部測定値をホールドします.
トリガ 2		タイミング入力が立ち下がり(OFF)エッジから次の立ち上がりエッジまでをサンプリング期間とします.	タイミング入力が立ち上がると, そのときから設定された平均回数分のデータをサンプリングして確定した内部測定値をホールドします.

使用例

```
' SetTriggerMode実行
Dim param As Variant
param = Array(0, 0)
Call caoCtrl.Execute("SetTriggerMode", param)
```

3.3.3.4.12. GetTriggerMode コマンド

トリガモードを取得します. 以下に引数と戻り値を示します.

項目	型説明	
引数	VT_I4	OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2
戻り値	VT_I4	トリガモードを取得します. 値の詳細は 3.3.3.4.11 を参照してください.

使用例

```
' GetTriggerMode実行
Dim mode As Integer
mode = caoCtrl.Execute("GetTriggerMode", 0)
```

3.3.3.4.13. SetOffset コマンド

オフセットを設定します. オフセットを設定することで, 表示値に任意の値を加算, 減算することができます. また, オフセットを設定しておくと, オートゼロを実行したときに, オフセット値を表

示できます. オフセット値は, 計測モード処理, オートゼロ処理を行った後の測定値に対して設定されます. 以下に引数を示します.

項目	型説明		
引数	VT_ARRAY | VT_I4		
	0	VT_I4	OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2
	1	VT_I4	オフセットを指定します. -999999～999999 の値を指定します.

使用例

```
' SetOffset実行
Dim param As Variant
param = Array(0, 0)
Call caoCtrl.Execute("SetOffset", param)
```

3.3.3.4.14. GetOffset コマンド

オフセットを取得します. 以下に引数と戻り値を示します.

項目	型説明	
引数	VT_I4	OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2
戻り値	VT_I4	オフセットを取得します.

使用例

```
' GetOffset実行
Dim offset As Integer
offset = caoCtrl.Execute("GetOffset", 0)
```

3.3.3.4.15. SetAnalogScaling コマンド

アナログ出力スケーリングを設定します.アナログ出力スケーリグの詳細につきましては,LK-G3000 マニュアル内の「3 章 -測定値の出力条件を設定する - アナログ出力をスケーリングする」を参照してください. 以下に引数を示します.

項目	型説明
引数	VT_ARRAY | VT_I4

項目	型説明		
	0	VT_I4	OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2
	1	VT_I4	入力値 1 を指定します. -999999～999999 の値を指定します.
	2	VT_I4	出力電圧値 1 を指定します. -10500～10500 の値を指定します.
	3	VT_I4	入力値 2 を指定します. -999999～999999 の値を指定します.
	4	VT_I4	出力電圧値 2 を指定します. -10500～10500 の値を指定します.

※ 以下の条件を満たさない場合, 設定に失敗し, API エラーが返ってきます.

(1) 入力値 1 - 入力値 2 ≠ 0

使用例

```
' SetAnalogScaling実行
Dim param As Variant
param = Array(0, 500, 10, 1000, 1000)
Call caoCtrl.Execute("SetAnalogScaling", param)
```

3.3.3.4.16. GetAnalogScaling コマンド

アナログ出力スケーリングを取得します. 以下に引数と戻り値を示します.

項目	型説明		
引数	VT_I4		OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2
戻り値	VT_ARRAY | VT_I4		
	0	VT_I4	入力値 1 を取得します.
	1	VT_I4	出力電圧値 1 を取得します.
	2	VT_I4	入力値 2 を取得します.
	3	VT_I4	出力電圧値 2 を取得します.

使用例

```
' GetAnalogScaling実行
Dim value As Variant
value = caoCtrl.Execute("GetAnalogScaling", 0)
If Not IsEmpty(value) Then
        ' 入力値1
```

```
        Dim inputValue1 As Integer
        inputValue1 = value(0)
        ' 出力電圧値1
        Dim outputValue1 As Integer
        outputValue1 = value(1)
        ' 入力値2
        Dim inputValue2 As Integer
        inputValue2 = value(2)
        ' 出力電圧値2
        Dim outputValue2 As Integer
        outputValue2 = value(3)
End If
```

3.3.3.4.17. SetCalcMode コマンド

計測モードを設定します. 計測モードの詳細については, LK-G3000 マニュアル内の「3 章 -測定値の出力条件を設定する - ホールド機能を使う(計測モード)」を参照してください. 以下に引数を示します.

項目	型説明		
引数	VT_ARRAY | VT_I4		
	0	VT_I4	OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2
	1	VT_I4	計測モードを指定します. 指定可能な計測モードの詳細については表 3-9 を参照してください. 以下のいずれかを指定してください. ・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

表 3-9 計測モード詳細

モード	機能
標準	測定した結果を随時表示/出力できます.
ピークホールド	サンプリング期間内の最大値を測定できます.
ボトムホールド	サンプリング期間内の最小値を測定できます.
ピーク to ピークホールド	サンプリング期間内の「最大値-最小値」を測定できます.
サンプルホールド	タイミング入力を ON にした瞬間の値を測定できます.

モード	機能
アベレージホールド	サンプリング期間内の平均値を測定できます.

使用例

```
' SetCalcMode実行
Dim param As Variant
param = Array(0, 0)
Call caoCtrl.Execute("SetCalcMode", param)
```

3.3.3.4.18. GetCalcMode コマンド

計測モードを取得します. 以下に引数と戻り値を示します.

項目	型説明	
引数	VT_I4	OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2
戻り値	VT_I4	計測モードを取得します. 値の詳細は 3.3.3.4.17 を参照してください.

使用例

```
' GetCalcMode実行
Dim mode As Integer
mode = caoCtrl.Execute("GetCalcMode", 0)
```

3.3.3.4.19. SetDisplayUnit コマンド

パネルに表示させる最小表示単位を設定します. 以下に引数を示します.

項目	型説明		
引数	VT_ARRAY | VT_I4		
	0	VT_I4	OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2
	1	VT_I4	最小表示単位を指定します. 以下のいずれかを指定してください. ・0 - 0.01mm ・1 - 0.001mm ・2 - 0.0001mm ・3 - 0.00001mm ・4 - 0.01μm ・5 - 0.001μm

使用例

```
' SetDisplayUnit実行
Dim param As Variant
param = Array(0, 0)
Call caoCtrl.Execute("SetDisplayUnit", param)
```

3.3.3.4.20. GetDisplayUnit コマンド

最小表示単位を取得します．以下に引数と戻り値を示します．

項目	型説明	
引数	VT_I4	OUT 番号を指定します．以下のいずれかを指定してください． ・0 - OUT1 ・1 - OUT2
戻り値	VT_I4	最小表示単位を取得します．値の詳細は 3.3.3.4.19 を参照してください．

使用例

```
' GetDisplayUnit実行
Dim unit As Integer
unit = caoCtrl.Execute("GetDisplayUnit", 0)
```

3.3.3.4.21. SetAnalogThrough コマンド

アナログスルーを設定します．計測モードで測定値をホールドしている場合に，アナログスルーを ON に設定すると，ホールドする前の内部測定値をアナログ出力します．以下に引数を示します．

項目	型説明		
引数	VT_ARRAY | VT_VARIANT		
	0	VT_I4	OUT 番号を指定します．以下のいずれかを指定してください． ・0 - OUT1 ・1 - OUT2
	1	VT_BOOL	アナログスルーを指定します． ・TRUE - ON ・FALSE - OFF

使用例

```
' SetAnalogThrough実行
Dim param As Variant
param = Array(0, false)
Call caoCtrl.Execute("SettAnalogThrough", param)
```

3.3.3.4.22. GetAnalogThrough コマンド

アナログスルーを取得します. 以下に引数と戻り値を示します.

項目	型説明	
引数	VT_I4	OUT 番号を指定します. 以下のいずれかを指定してください. ・0 - OUT1 ・1 - OUT2
戻り値	VT_BOOL	アナログスルーを取得します.

使用例

```
' GetAnalogThrough実行
Dim value As Boolen
value = caoCtrl.Execute("GetAnalogThrough", 0)
```

3.3.3.5. 共通設定関連コマンド

ヘッド設定と OUT 設定に関連する共通機能を設定/取得するコマンドです.

3.3.3.5.1. SetDataStorage コマンド

データストレージの蓄積させる対象 OUT, 蓄積点数, 蓄積周期を設定します. 以下に引数を示します.

項目	型説明		
引数	VT_ARRAY | VT_I4		
	0	VT_I4	対象 OUT を指定します. 以下のいずれかを指定してください. ・0 - 対象 OUT なし ・1 - OUT1 ・2 - OUT2 ・3 - OUT1 と OUT2
	1	VT_I4	蓄積点数を指定します. 1～65536 の値を指定します.

項目	型説明		
	2	VT_I4	蓄積周期を指定します. 以下のいずれかを指定してください. ・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

```
' SetDataStorage実行
Dim param As Variant
param = Array(1, 100, 0)
Call caoCtrl.Execute("SetDataStorage", param)
```

3.3.3.5.2. GetDataStorage コマンド

データストレージの対象 OUT, 蓄積点数, 蓄積周期を取得します. 以下に引数と戻り値を示します.

項目	型説明		
戻り値	VT_ARRAY | VT_I4		
	0	VT_I4	対象 OUT を取得します. 値の詳細は SetDataStorage コマンドを参照してください.
	1	VT_I4	蓄積点数を取得します.
	2	VT_I4	蓄積周期を取得します. 値の詳細は SetDataStorage コマンドを参照してください.

使用例

```
' GetDataStorage実行
Dim value As Variant
value = caoCtrl.Execute("GetDataStorage")
If Not IsEmpty(value) Then
        ' 対象OUT
        Dim out As Integer
        out = value(0)
        ' 蓄積点数
        Dim dataCnt As Integer
```

```
        dataCnt = value(1)
        ' 蓄積周期
        Dim cycle As Integer
        cycle = value(2)
End If
```

3.3.3.5.3. SetSamplingCycle コマンド

サンプリング周期を設定します. 以下に引数を示します.

項目	型説明	
引数	VT_I4	サンプリング周期を指定します. 以下のいずれかを指定してください. ・0 - 20μs ・1 - 50μs ・2 - 100μs ・3 - 200μs ・4 - 500μs ・5 - 1000μs

使用例

```
' SetSamplingCycle実行
Call caoCtrl.Execute("SetSamplingCycle", 0)
```

3.3.3.5.4. GetSamplingCycle コマンド

サンプリング周期を取得します. 以下に戻り値を示します.

項目	型説明	
戻り値	VT_I4	サンプリング周期を取得します. 値の詳細は 3.3.3.5.3 を参照してください.

使用例

```
' GetSamplingCycle実行
Dim value As Integer
value = caoCtrl.Execute("GetSamplingCycle")
```

3.3.3.5.5. SetMutualInterPrev コマンド

相互干渉防止を設定します. 相互干渉防止を ON にすると 2 台のヘッドが交互に発行して相手側ヘッドの干渉を受けなくなります. 以下に引数を示します.

項目	型説明

引数	VT_BOOL	相互干渉防止を指定します. ・TRUE - ON ・FALSE - OFF

使用例

```
' SetMutualInterPrev実行
Call caoCtrl.Execute("SetMutualInterPrev", true)
```

3.3.3.5.6. GetMutualInterPrev コマンド

相互干渉防止を取得します. 以下に戻り値を示します.

項目	型説明	
戻り値	VT_BOOL	相互干渉防止を取得します.

使用例

```
' GetMutualInterPrev実行
Dim value As Boolen
value = caoCtrl.Execute("GetMutualInterPrev")
```

3.3.3.5.7. SetTimingSync コマンド

OUT1 と OUT2 のタイミング入力の制御方法を設定します. 以下に引数を示します.

項目	型説明	
引数	VT_I4	タイミング同期を指定します. タイミング同期の詳細については, 表 3-10 を参照してください. 以下のいずれかを指定してください. ・0 - 非同期 ・1 - 同期

表 3-10 タイミング同期詳細

タイミング同期	機能
非同期	OUT1 と OUT2 を非同期で制御します. OUT1, OUT2 それぞれに独立した入力端子を割り当てます. ・OUT1 : 12 極端子台の 8 番 ・OUT2 : 拡張コネクタの 8 番
同期	OUT1 と OUT2 を同期して制御します. 12 極端子台の 8 番が対応し, 拡張コネクタの 6 番は無効になります.

※ 入力端子に関しましては, LK-G3000 マニュアル内の「4 章 入力端子」を参照してください.

使用例

```
' SetTimingSync実行
Call caoCtrl.Execute("SetTimingSync", 0)
```

3.3.3.5.8. GetTimingSync コマンド

タイミング同期を取得します. 以下に戻り値を示します.

項目	型説明	
戻り値	VT_I4	タイミング同期を取得します. 値の詳細は 3.3.3.5.7 を参照してください.

使用例

```
' GetTimingSync実行
Dim value As Integer
value = caoCtrl.Execute("GetTimingSync")
```

3.3.3.5.9. SetToleCompOutputFormat コマンド

公差判定出力の出力形態を設定します. 以下に引数を示します.

項目	型説明	
引数	VT_I4	判定出力形態を指定します. 判定出力形態の詳細については, 表 3-11 を参照してください. 以下のいずれかを指定してください. ・0 - ノーマル ・1 - ホールド ・2 - オフディレイ

表 3-11 判定出力形態詳細

タイミング同期	機能
ノーマル	公差判定に合わせて出力します.
ホールド	ON になった出力をホールドします. 測定値リセットでホールドを解除します.
オフディレイ	ノーマル出力に 60ms のオフディレイがかかります. 測定値リセットでホールドを解除します.

使用例

```
' SetToleCompOutputFormat実行
Call caoCtrl.Execute("SetToleCompOutputFormat", 0)
```

3.3.3.5.10. GetToleCompOutputFormat コマンド

公差判定出力の出力形態を取得します. 以下に戻り値を示します.

項目	型説明	
戻り値	VT_I4	判定出力形態を取得します. 値の詳細は 3.3.3.5.9 を参照してください.

使用例

```
' GetToleCompOutputFormat実行
Dim value As Integer
value = caoCtrl.Execute("GetToleCompOutputFormat")
```

3.3.3.5.11. SetStorobeTime コマンド

ストローブ出力が ON する時間(ワンショット出力時間)を設定します. 以下に引数を示します.

項目	型説明	
引数	VT_I4	ストローブ時間を指定します. 以下のいずれかを指定してください. ・0 - 2ms ・1 - 5ms ・2 - 10ms ・3 - 20ms

使用例

```
' SetStorobeTime実行
Call caoCtrl.Execute("SetStorobeTime", 0)
```

3.3.3.5.12. GetStorobeTime コマンド

ストローブ出力が ON する時間(ワンショット出力時間)を取得します. 以下に戻り値を示します.

項目	型説明	
戻り値	VT_I4	ストローブ時間を取得します. 値の詳細は 3.3.3.5.11 を参照してください.

使用例

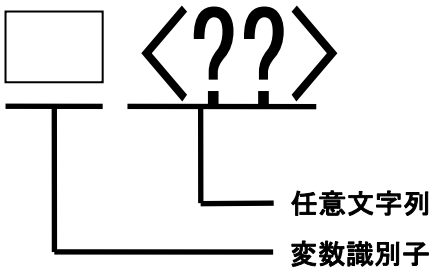
```
' GetStorobeTime実行
Dim value As Integer
value = caoCtrl.Execute("GetStorobeTime")
```

3.4. 変数一覧

各クラスで使用可能な変数一覧を定義します. なお変数は, CaoVariable クラスのオブジェクトを指します. 複数変数を登録（オプションのみ変更したい場合等に有用）するために任意の文字列を付与することが可能です.

変数名に任意文字列を付与するための書式を以下に示します.

複数変数共通指定書式



3.4.1. CaoController クラス変数

変数名	説明	Value		参照
		get	put	
@MAKER_NAME	メーカー名を取得します.	○	-	P.57
@VERSION	DLL バージョンを取得します.	○	-	P.57
@CALCDATA	測定値を取得します.	○	-	P.58
RECEIVED_WAVEFROM<??>	受光波形を取得します.	○	-	P.59

3.4.1.1. @MAKER_NAME

メーカー名の取得をします.

データ型

型説明	
VT_BSTR	メーカー名を取得します.

使用例

```
' 変数追加
Dim var As CaoVariable
var = caoCtrl.AddVariable("@MAKER_NAME")
' 値取得
Dim name As String
name = var.value
```

3.4.1.2. @VERSION

DLL のバージョンの取得をします.

データ型

型説明	
VT_BSTR	DLL のバージョンを取得します. *.*.*

使用例

```
' 変数追加
Dim var As CaoVariable
var = caoCtrl.AddVariable("@VERSION ")
' 値取得
Dim version As String
version = var.value
```

3.4.1.3. @CALCDATA

測定値の取得をします.

データ型

型説明			
VT_ARRAY | VT_VARIANT			
0	VT_ARRAY | VT_VARIANT		OUT1 の測定値
	0	VT_I4	有効データかどうか. 以下のいずれかが取得されます. ・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
	1	VT_R4	測定値. 有効データ以外の場合は無効な値になります.
1	VT_ARRAY | VT_VARIANT		OUT2 の測定値
	0	VT_I4	有効データかどうか. 上記の OUT1 の測定値と同様の値を取得します.
	1	VT_R4	測定値. 有効データ以外の場合は無効な値になります.

使用例

```
' 変数追加
Dim var As CaoVariable
Set var = caoCtrl.AddVariable("@CALCDATA")
' 値の取得
Dim values As Variant
values = var.Value

If Not IsEmpty(values) Then
        ' OUT1の測定値
        Dim value1 As Variant
        value1 = values(0)
```

```
        ' 有効データかどうか
        Dim validData1 As Long
        validData1 = value1(0)
        '測定値
        Dim fValue1 As Single
        fValue1 = value1(1)

        'OUT2の測定値
        Dim value2 As Variant
        value2 = values(1)
        '有効データかどうか
        Dim validData2 As Long
        validData2 = value2(0)
        ' 測定値
        Dim fValue2 As Single
        fValue2 = value2(1)
End If
```

3.4.1.4. RECEIVED_WAVEFROM<??>

受光波形を取得します. RECEIVED_WAVEFROM の後に任意の文字列を入力して変数名を指定してください.

オプション

オプション	必須	説明	値範囲	デフォルト値
HeadNo	--	取得したいヘッド番号を指定します. 0(HEAD-A),1(HEAD-B)を指定します.	0 - 1	0
PeekNo	--	測定モードが「透明体 2」の時, 波形を取得するピーク位置を指定します.	0 - 3	0

データ型

型説明			
VT_ARRAY | VT_VARIANT			
0	VT_ARRAY | VT_I4		測定位置
	0	VT_I4	ピーク番号 0 の測定位置
	1	VT_I4	ピーク番号 1 の測定位置
	2	VT_I4	ピーク番号 2 の測定位置
	3	VT_I4	ピーク番号 3 の測定位置
1	VT_I4		有効データ数
2	VT_ARRAY | VT_UI1		読み出したデータ
	0	VT_I4	1 番目のデータ

型説明			
	…	…	…
	1023	VT_I4	1024 番目のデータ

使用例

```
' 変数追加
Dim var As CaoVariable
Set var = caoCtrl.AddVariable("RECEIVED_WAVEFROM1","HeadNo = 0,PeekNo = 0")

' 値の取得
Dim value As Variant
value = var.Value
If Not IsEmpty(value) Then
        ' 測定位置
        Dim mesurePosition As Variant
        mesurePosition = value(0)
        If Not IsEmpty(value) Then
                ' ピーク番号0の測定位置
                Dim peekPosition0 As Integer
                peekPosition0 = mesurePosition(0)
                ' ピーク番号1の測定位置
                Dim peekPosition1 As Integer
                peekPosition1 = mesurePosition(1)
                ' ピーク番号2の測定位置
                Dim peekPosition2 As Integer
                peekPosition2 = mesurePosition(2)
                ' ピーク番号3の測定位置
                Dim peekPosition3 As Integer
                peekPosition3 = mesurePosition(3)
        End If

        ' 有効データ数
        Dim effDataCnt As Integer
        effDataCnt = value(1)

        ' 読み出したデータ
        Dim allReadData As Variant
        allReadData = value(2)
        If Not IsEmpty(value) Then
                Dim i As Integer
                For i = 0 To 1023
                        Dim readData As Byte
                        readData = allReadData(i)
                Next i
        End If
End If
```

4. LK-G3000LkIF プロバイダによるプログラミング

LK-G3000LkIF プロバイダでは，以下の手順で機器との通信の準備を行います.

- CaoEngine の作成
- CaoWorkspace の作成
- CaoController の作成

手順を実行した後は，CaoController の Execute メソッドを使用する，もしくは，CaoVariabe オブジェクトの Value プロパティの取得を行うことで，LK-G3000 シリーズとの通信を行い機器の情報にアクセスすることができます.

4.1. OUT1 と OUT2 の測定値を取得するサンプルプログラミング

ここでは例として OUT1 と OUT2 の測定値を読み込むサンプルプログラムを示します. 表 4-1 にサンプルプログラムの要件を，図 4-1 にサンプルプログラムの流れをそれぞれ記述しています.

表 4-1 サンプルプログラムの要件

要件	説明
処理内容	LK-G3000 から OUT1 及び OUT2 の測定値を読込む.

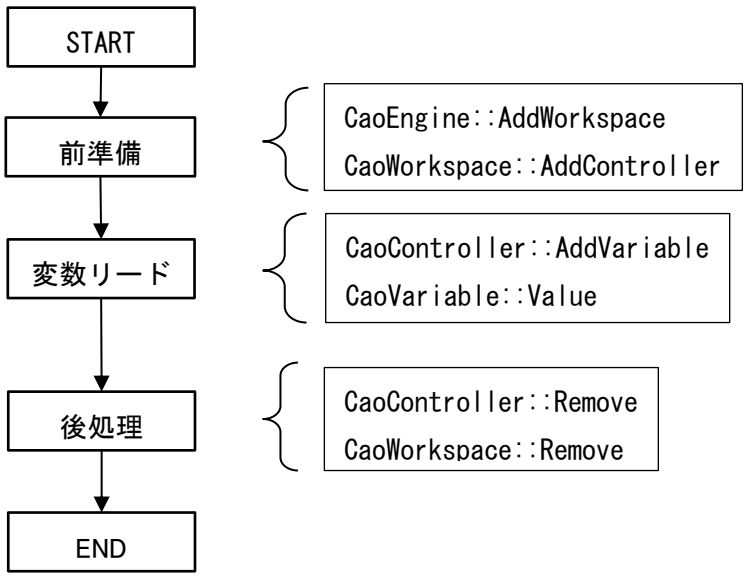


図 4-1 OUT1 及び OUT2 の測定値読み込みの流れ

以降の節から具体的なコードを示します.

4.1.1. サンプルプログラム

以下にサンプルプログラムの全体像を示します.

Sample GetCalcData.vb

```
' オブジェクト
Dim caoEng As CaoEngine
```

```
Dim caoWs As CaoWorkspace
Dim caoCtrl As CaoController
Dim caoVrl As CaoVariable

Private Sub Main()
    ' 準備
    Call Preprocessing

    ' 値の取得
    Dim values As Variant
    values = caoVrl.value

    If Not IsEmpty(values) Then
        ' OUT1の測定値
        Dim value1 As Variant
        value1 = values(0)
        ' 有効データかどうか
        Dim validData1 As Long
        validData1 = value1(0)
        ' 測定値
        Dim fValue1 As Single
        fValue1 = value1(1)

        ' OUT2の測定値
        Dim value2 As Variant
        value2 = values(1)
        ' 有効データかどうか
        Dim validData2 As Long
        validData2 = value2(0)
        ' 測定値
        Dim fValue2 As Single
        fValue2 = value2(1)
    End If

    ' 後処理
    Call Postprocessing
End Sub

' 準備メソッド
Private Sub Preprocessing()
    ' CaoEngine オブジェクトの生成
    Set caoEng = New CaoEngine
    ' CaoWorkspace オブジェクトの生成
    Set caoWs = caoEng.AddWorkspace("Workspace", "")
    ' CaoController オブジェクトの生成
    Set caoCtrl = caoWs.AddController("LKG3000LkIF", _
                                      "CaoProv.KEYENCE.LK-G3000LkIF", _
                                      "", _
```

```
                                        "Timeout=1000")
    ' CaoVariable オブジェクトの生成
    Set caoVrl = caoCtrl.AddVariable("@CALCDATA", "")
End Sub

' 後処理メソッド
Private Sub Postprocessing()
    ' CaoController からCaoVariable を削除
    Call caoCtrl.variables.Remove(caoVrl.Index)
    ' CaoVariableの消去
    Set caoVrl = Nothing
    ' CaoWorkspace からCaoController を削除
    Call caoWs.Controllers.Remove(caoCtrl.Index)
    ' CaoController の消去
    Set caoCtrl = Nothing
    ' CaoEngine からCaoWorkspace を削除
    Call caoEng.Workspaces.Remove(caoWs.Index)
    ' CaoWorkspace の消去
    Set caoWs = Nothing
    ' CaoEngine の消去
    Set caoEng = Nothing
End Sub
```

4.1.1.1. 前処理

機器と通信するためには，以下の手順を取ります．

(1) オブジェクトを保持するための変数を用意します．コントローラ接続に必要なオブジェクトは，CaoEngineオブジェクトとCaoWorkspaceオブジェクトとCaoControllerオブジェクトです．CaoWorkpaceオブジェクトは，CaoControllerオブジェクトをCaoWorkspacesから取得する場合には変数を用意する必要はありません．また変数にアクセスするためのCaoVariableオブジェクトも必要になります．以下にVB6でのコード例を示します．

```
' CaoEngine オブジェクト用の変数
Dim caoEng As CaoEngine
' CaoWorkspace オブジェクト用の変数
Dim caoWs As CaoWorkspace
' CaoController オブジェクト用の変数
Dim caoCtrl As CaoController
' CaoVariable オブジェクト用の変数
Dim caoVrl As CaoVariable
```

(2) CaoEngineオブジェクトを生成します．CaoEngineオブジェクトはNewキーワードを使って生成します．

```
    ' CaoEngine オブジェクトの生成
    Set caoEng = New CaoEngine
```

(3) CaoWorkspaceオブジェクトを取得もしくは生成します. CaoEngineオブジェクトを生成すると, デフォルトでCaoWorkspacesオブジェクトとCaoWorkspaceオブジェクトを1つずつ生成しています. 以下にCaoWorkspaceオブジェクトを新しく生成するコード例とデフォルトのCaoWorkspaceを示します.

```
' CaoWorkspace オブジェクトの生成
Set caoWs = caoEng.AddWorkspace("Workspace", "")
```

(4) CaoControllerオブジェクトを生成します. CaoControllerオブジェクトを生成するには, 使用するプロバイダ名と使用するためのパラメータを設定します. LK-G3000LkIFプロバイダでは, APIからの応答時間をオプションで指定します. 以下にコード例を示します.

```
' CaoController オブジェクトの生成
Set caoCtrl = caoWs.AddController("LKG3000LkIF", _
                                  "CaoProv.KEYENCE.LK-G3000LkIF", _
                                  "" _
                                  , _
                                  "Timeout=1000")
```

(5) CaoVariableオブジェクトを生成します.取得したい変数のCaoVariableオブジェクトを生成します.以下にOUT1とOUT2の測定値にアクセスする変数オブジェクトを生成するコード例を示します.

```
' CaoVariable オブジェクトの生成
Set caoVrl = caoCtrl.AddVariable("@CALCDATA", "")
```

4.1.1.2. OUT1 と OUT2 の測定値取得

OUT1 と OUT2 の測定値を取得するには,CaoVariable オブジェクトの Value プロパティを参照します. 測定値ごとに変数を用意する必要があります. 以下にコード例を示します.

```
' 値の取得
Dim values As Variant
values = caoVrl.value

If Not IsEmpty(values) Then
        ' OUT1の測定値
        Dim value1 As Variant
        value1 = values(0)
        ' 有効データかどうか
        Dim validData1 As Long
        validData1 = value1(0)
        ' 測定値
        Dim fValue1 As Single
        fValue1 = value1(1)

        ' OUT2の測定値
        Dim value2 As Variant
        value2 = values(1)
```

```
        ' 有効データかどうか
        Dim validData2 As Long
        validData2 = value2(0)
        ' 測定値
        Dim fValue2 As Single
        fValue2 = value2(1)
End If
```

4.1.1.3. 後処理

後処理を行う場合には，生成したオブジェクトを消去すると共に，オブジェクトを管理するコレクションクラスから消去するオブジェクトを削除します．以下にコード例を示します．

```
' CaoController から CaoVariable を削除
Call caoCtrl.variables.Remove(caoVrl.Index)
' CaoVariable の消去
Set caoVrl = Nothing
' CaoWorkspace から CaoController を削除
Call caoWs.Controllers.Remove(caoCtrl.Index)
' CaoController の消去
Set caoCtrl = Nothing
' CaoEngine から CaoWorkspace を削除
Call caoEng.Workspaces.Remove(caoWs.Index)
' CaoWorkspace の消去
Set caoWs = Nothing
' CaoEngine の消去
Set caoEng = Nothing
```

4.2. データストレージの蓄積データを取得するサンプルプログラミング

ここでは例としてデータストレージに蓄積されているデータを読み込むサンプルプログラムを示します．表 4-2 にサンプルプログラムの要件を，図 4-2 にサンプルプログラムの流れをそれぞれ記述しています．

表 4-2 サンプルプログラムの要件

要件	説明
処理内容	LK-G3000 からデータストレージの蓄積データを読込む．

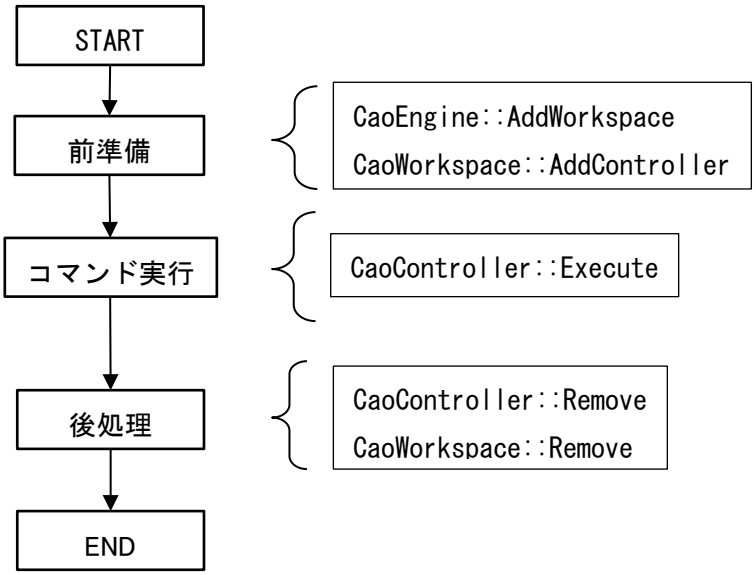


図 4-2 OUT1 及び OUT2 の測定値読み込みの流れ

以降の節から具体的なコードを示します.

4.2.1. サンプルプログラム

以下にサンプルプログラムの全体像を示します.

Sample GetDataStorageData.vb

```
' オブジェクト
Dim caoEng As CaoEngine
Dim caoWs As CaoWorkspace
Dim caoCtrl As CaoController

Private Sub Main()
    ' 準備
    Call Preprocessing

    ' 引数の指定
    Dim param As Variant
    ' 要素1にOUT番号, 要素2に取得したい蓄積データ個数を指定
    param = Array(0, 3)
    ' 値の取得
    Dim value As Variant
    value = caoCtrl.Execute("GetDataStorageData", param)

    If Not IsEmpty(value) Then
        Dim i As Integer
        ' 取得した蓄積データ数分繰り返す
        For i = LBound(value) To (UBound(value) - 1)
            ' 蓄積データ
            Dim accumulationData As Variant
            accumulationData = value(i)
```

```
            If Not IsEmpty(accumulationData) Then
                ' 有効データかどうか
                Dim result As Integer
                result = accumulationData(LBound(accumulationData))
                ' 測定値
                Dim fValue As Single
                fValue = accumulationData(UBound(accumulationData))
            End If
        Next i
        ' 取得した蓄積データ数
        Dim readDataCnt As Integer
        readDataCnt = value(UBound(value))
    End If

    ' 後処理
    Call Postprocessing
End Sub

' 準備メソッド
Private Sub Preprocessing()
    ' CaoEngine オブジェクトの生成
    Set caoEng = New CaoEngine
    ' CaoWorkspace オブジェクトの生成
    Set caoWs = caoEng.AddWorkspace("Workspace", "")
    ' CaoController オブジェクトの生成
    Set caoCtrl = caoWs.AddController("LKG3000LkIF", _
                                      "CaoProv.KEYENCE.LK-G3000LkIF", _
                                      "", _
                                      "Timeout=1000")
End Sub

' 後処理メソッド
Private Sub Postprocessing()
    ' CaoWorkspace からCaoController を削除
    Call caoWs.Controllers.Remove(caoCtrl.Index)
    ' CaoController の消去
    Set caoCtrl = Nothing
    ' CaoEngine からCaoWorkspace を削除
    Call caoEng.Workspaces.Remove(caoWs.Index)
    ' CaoWorkspace の消去
    Set caoWs = Nothing
    ' CaoEngine の消去
    Set caoEng = Nothing
End Sub
```

4.2.1.1. 前処理

機器と通信するためには，以下の手順を取ります.

(1) オブジェクトを保持するための変数を用意します．コントローラ接続に必要なオブジェクトは，CaoEngineオブジェクトとCaoWorkspaceオブジェクトとCaoControllerオブジェクトです．CaoWorkpaceオブジェクトは，CaoControllerオブジェクトをCaoWorkspacesから取得する場合には変数を用意する必要はありません．以下にVB6でのコード例を示します．

```
' CaoEngine オブジェクト用の変数
Dim caoEng As CaoEngine
' CaoWorkspace オブジェクト用の変数
Dim caoWs As CaoWorkspace
' CaoController オブジェクト用の変数
Dim caoCtrl As CaoController
```

(2) CaoEngineオブジェクトを生成します．CaoEngineオブジェクトはNewキーワードを使って生成します．

```
    ' CaoEngine オブジェクトの生成
    Set caoEng = New CaoEngine
```

(3) CaoWorkspaceオブジェクトを取得もしくは生成します．CaoEngineオブジェクトを生成すると，デフォルトでCaoWorkspacesオブジェクトとCaoWorkspaceオブジェクトを1つずつ生成しています．以下にCaoWorkspaceオブジェクトを新しく生成するコード例とデフォルトのCaoWorkspaceを示します．

```
    ' CaoWorkspace オブジェクトの生成
    Set caoWs = caoEng.AddWorkspace("Workspace", "")
```

(4) CaoControllerオブジェクトを生成します．CaoControllerオブジェクトを生成するには，使用するプロバイダ名と使用するためのパラメータを設定します．LK-G3000LkIFプロバイダでは，APIからの応答時間をオプションで指定します．以下にコード例を示します．

```
    ' CaoController オブジェクトの生成
    Set caoCtrl = caoWs.AddController("LKG3000LkIF", _
                                      "CaoProv.KEYENCE.LK-G3000LkIF", _
                                      "", _
                                      "Timeout=1000")
```

4.2.1.2. データストレージの蓄積データ取得

データストレージの蓄積データを取得するには，CaoController オブジェクトの Execute メソッドを実行します．以下にコード例を示します．

```
' 引数の指定
Dim param As Variant
' 要素1にOUT番号，要素2に取得したい蓄積データ個数を指定
param = Array(0, 3)
' 値の取得
Dim value As Variant
```

```
value = caoCtrl.Execute("GetDataStorageData", param)

If Not IsEmpty(value) Then
        Dim i As Integer
        ' 取得した蓄積データ数分繰り返す
        For i = LBound(value) To (UBound(value) - 1)
                ' 蓄積データ
                Dim accumulationData As Variant
                accumulationData = value(i)
                If Not IsEmpty(accumulationData) Then
                        ' 有効データかどうか
                        Dim result As Integer
                        result = accumulationData(LBound(accumulationData))
                        ' 測定値
                        Dim fValue As Single
                        fValue = accumulationData(UBound(accumulationData))
                End If
        Next i
        ' 取得した蓄積データ数
        Dim readDataCnt As Integer
        readDataCnt = value(UBound(value))
End If
```

4.2.1.3. 後処理

後処理を行う場合には，生成したオブジェクトを消去すると共に，オブジェクトを管理するコレクションクラスから消去するオブジェクトを削除します．以下にコード例を示します．

```
' CaoWorkspace から CaoController を削除
Call caoWs.Controllers.Remove(caoCtrl.Index)
' CaoController の消去
Set caoCtrl = Nothing
' CaoEngine から CaoWorkspace を削除
Call caoEng.Workspaces.Remove(caoWs.Index)
' CaoWorkspace の消去
Set caoWs = Nothing
' CaoEngine の消去
Set caoEng = Nothing
```

5. LK-G3000LkIF プロバイダエラーコード

本プロバイダには，独自エラーコードは存在しませんが，API が異常終了した場合のエラーコードが存在します．（表 5-1 エラーコード表参照）

ORiN2 の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 5-1 エラーコード表

エラー番号	説明
0x80100001	APIが異常終了した場合

6. 付録

付録A. API 対応表

CaoController::Execute

コマンド名	API 関数名
SetMode	LKIF_SetMode
GetCalcData	LKIF_GetCalcData
SetTiming	LKIF_SetTiming
SetZero	LKIF_SetZero
SetReset	LKIF_SetReset
SetPanelLock	LKIF_SetPanelLock
SetProgramNo	LKIF_SetProgramNo
GetProgramNo	LKIF_GetProgramNo
GetFigureData	LKIF_GetFigureData
ClearFigureData	LKIF_ClearFigureData
StartDataStorage	LKIF_DataStorageStart
StopDataStorage	LKIF_DataStorageStop
ClearDataStorage	LKIF_DataStorageInit
GetDataStorageData	LKIF_DataStorageGetData
GetDataStorageStatus	LKIF_DataStorageGetStatus
GetLight	LKIF_GetLight
SetPanel	LKIF_SetPanel
GetPanel	LKIF_GetPanel
SetTolerance	LKIF_SetTolerance
GetTolerance	LKIF_GetTolerance
SetAbleMode	LKIF_SetAbleMode
GetAbleMode	LKIF_GetAbleMode
SetAbleMinMax	LKIF_SetAbleMinMax
GetAbleMinMax	LKIF_GetAbleMinMax
SetMeasureMode	LKIF_SetMeasureMode
GetMeasureMode	LKIF_GetMeasureMode
SetNumAlarm	LKIF_SetNumAlarm
GetNumAlarm	LKIF_GetNumAlarm
SetAlarmLevel	LKIF_SetAlarmLevel
GetAlarmLevel	LKIF_GetAlarmLevel

StartABLE	LKIF_AbleStart
StopABLE	LKIF_AbleStop
CancelABLE	LKIF_AbleCancel
SetReflectionMode	LKIF_SetReflectionMode
GetReflectionMode	LKIF_GetReflectionMode
SetCalcMethod	LKIF_SetCalcMethod
GetCalcMethod	LKIF_GetCalcMethod
SetScaling	LKIF_SetScaling
GetScaling	LKIF_GetScaling
SetFilterMode	LKIF_SetFilterMode
GetFilterMode	LKIF_GetFilterMode
SetAverage	LKIF_SetAverage
GetAverage	LKIF_GetAverage
SetCutOffFrequency	LKIF_SetCutOffFrequency
GetCutOffFrequency	LKIF_GetCutOffFrequency
SetTriggerMode	LKIF_SetTriggerMode
GetTriggerMode	LKIF_GetTriggerMode
SetOffset	LKIF_SetOffset
GetOffset	LKIF_GetOffset
SetAnalogScaling	LKIF_SetAnalogScaling
GetAnalogScaling	LKIF_GetAnalogScaling
SetCalcMode	LKIF_SetCalcMode
GetCalcMode	LKIF_GetCalcMode
SetDisplayUnit	LKIF_SetDisplayUnit
GetDisplayUnit	LKIF_GetDisplayUnit
SetAnalogThrough	LKIF_SetAnalogThrough
GetAnalogThrough	LKIF_GetAnalogThrough
SetDataStorage	LKIF_SetDataStorage
GetDataStorage	LKIF_GetDataStorage
SetSamplingCycle	LKIF_SetSamplingCycle
GetSamplingCycle	LKIF_GetSamplingCycle
SetMutualInterPrev	LKIF_SetMutualInterferencePrevention
GetMutualInterPrev	LKIF_GetMutualInterferencePrevention
SetTimingSync	LKIF_SetTimingSynchronization
GetTimingSync	LKIF_GetTimingSynchronization

SetToleCompOutputFormat	LKIF_SetToleranceComparatorOutputFormat
GetToleCompOutputFormat	LKIF_GetToleranceComparatorOutputFormat
SetStorobeTime	LKIF_SetStorobeTime
GetStorobeTime	LKIF_GetStorobeTime

CaoVariable

変数名	Get_Value	Set_Value
@CALCDATA	LKIF_GetCalcData	---
RECEIVED_WAVEFROM <??>	LKIF_GetLight	---