KEYENCE
GT2DLEP1 プロバイダ

Version 1.0.1

ユーザーズ ガイド

July 11, 2017

備考:

【改版履歴】

バージョン	日付	内容
1.0.0	2017-3-16	初版.
1.0.1	2017-7-11	誤記修正 @PeekValue （誤）→@PeakValue（正） @JadgeDelay（誤）→@JudgeDelay（正）

【動作確認機種】

機種	バージョン	注意事項
GT2 シリーズ		通信ユニット(DL-EP1) + アンプユニット(GT2-71N 等) + センサユニット(GT2-H12 等) の構成

目次

1. はじめに

KEYENCE GT2DLEP1 プロバイダ(以下 GT2DLEP1 プロバイダ)は，KEYENCE 製接触式変位センサ(GT2 シリーズ)に対し，通信ユニット(DL-EP1)を通じてアクセスを行う ORiN2 CAO プロバイダです.

本ドキュメントでは，GT2DLEP1 プロバイダの概要と，実装されている CAO インタフェース（関数仕様）について説明しています.

2. プロバイダの概要

2.1. インストール

GT2DLEP1 プロバイダモジュールは，下記の DLL で構成されています．ORiN2 SDK のインストーラでインストールした場合は，インストール作業は不要です．手動でインストールする場合は，表 2-1 のように実行してください．

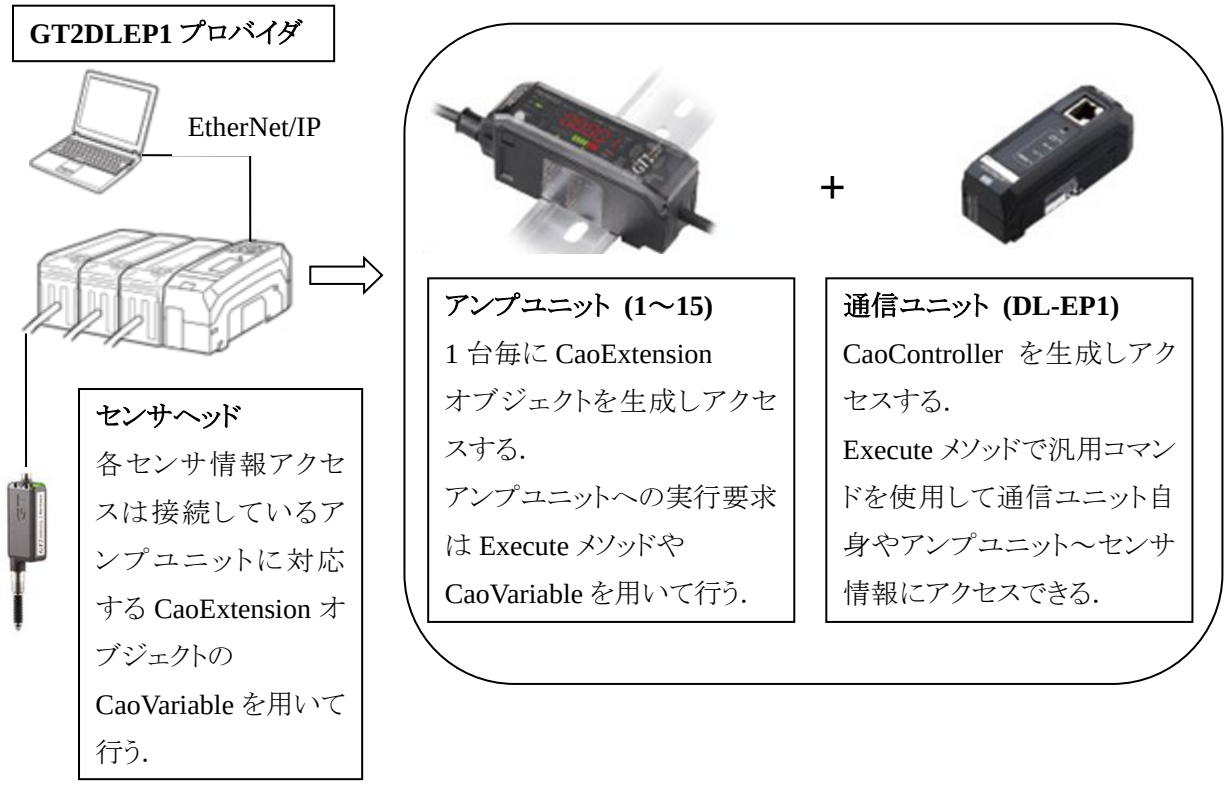
表 2-1 GT2DLEP1 プロバイダ

ファイル名	CaoProvKEYENCEGT2DLEP1.dll
ProgID	CaoProv.KEYENCE.GT2DLEP1
レジストリ登録	regsvr32 CaoProvKEYENCEGT2DLEP1.dll
レジストリ登録の抹消	regsvr32 /u CaoProvKEYENCEGT2DLEP1.dll

2.2. 概要

GT2DLEP1 プロバイダは，CIP プロトコルを用いて DE-EP1 へのアクセスを行います．

DE-EP1 に接続されたアンプユニット毎に CaoExtension オブジェクトを生成し，Execute メソッド又は CaoVariable への読み書きでセンサの情報へアクセスします．



3. メソッド・プロパティ

3.1. CaoWorkspace::AddController メソッド

GT2DLEP1 プロバイダは AddController 時に通信用の接続パラメータを参照し，通信の接続を行います．

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

bstrCtrlName : [in] コントローラ名
bstrProvName : [in] プロバイダ名．固定値 = ” CaoProv.KEYENCE.GT2DLEP1”
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 3-1 CaoWorkspace::AddController のオプション文字列

オプション ([1])	説明
Conn=<接続パラメータ>	必須． 通信形態と接続パラメータ． （参照 3.1.1）
MyIP[=<自 IP アドレス>]	自 IP アドレス．(複数 NIC 用途) （デフォルト：指定なし）
ConnTimeout [=<タイムアウト時間>]	TCP 接続自のタイムアウト時間．(ミリ秒) （デフォルト：3000）
TimeOut[=<タイムアウト時間>]	送受信時のタイムアウト時間．(ミリ秒) （デフォルト：3000）

3.1.1. Conn オプション

通信形態と接続パラメータを指定します．
以下に Conn オプションの接続パラメータ文字列を示します．

Ethernet/IP デバイス

“Conn=ETH:<Dest IP Address>[:<Dest Port No>[:<Src IP Address>[:<Src Port No>]]]”

< Dest IP Address > : TCP/IP 接続先 IP アドレス．
例：“127.0.0.1”，“192.168.0.1”

<Dest Port No> : TCP/IP 接続ポート番号．

[1] 角括弧(“[ ]”)内は省略可能を示します．また，各パラメータの解説中の下線部はオプションを指定しなかったときのデフォルト値になります．

例:44818, 5006, 5007

<Src IP Address> : 自 IP アドレス. (複数 NIC 用途) ([2])

<Src Port No> : 自ポート番号. (複数 NIC 用途)

3.2. CaoControllere::Execute メソッド

CaoController クラスの Execute メソッドは, コマンドを実行するためのメソッドです. 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 4.1 章を参照してください.

書式 [<vntRet:VARIANT> =] Execute(<bstrCmd:BSTR > [,<vntParam:VARIANT>])

< bstrCmd> : [in] コマンド名

< vntParam> : [in] パラメータ

3.3. CaoController::AddVariable メソッド

CaoController クラスの AddVariable メソッドは, 変数オブジェクトを作成するためのメソッドです. 使用できる変数一覧は 5.1 章を参照してください.

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

<bstrVariableName> : [in] 変数名

<bstrOption> : [in] オプション文字列

[2] Conn オプションと MyIP オプションの両方で自 IP アドレスを指定するとエラーになります. 利用する場合は必ずどちらか片方で指定するようにしてください.

3.4. CaoController::AddExtension メソッド

CaoController クラスの AddExtension メソッドは，DL-EP1 に接続されたアンプユニット毎の Extension オブジェクトを作成するためのメソッドです.

書式　AddExtension(<bstrChannelNo:VT_BSTR>)

< bstrChannelNo>　:　[in] アンプユニットの ID 番号.
“CH<ID 番号>”の書式で指定. (ID 番号:1～15)

3.5. CaoExtension::Execute メソッド

CaoExtension クラスの Execute メソッドは，コマンドを実行するためのメソッドです. 第1引数にコマンド名，第 2 引数にパラメータを指定します. 各コマンドの詳細は 4.2 章を参照してください.

書式　[<vntRet:VARIANT> =] Execute(<bstrCmd:BSTR > [,<vntParam:VARIANT>])

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

3.6. CaoExtension::AddVariable メソッド

CaoExtension クラスの AddVariable メソッドは，変数オブジェクトを作成するためのメソッドです. 使用できる変数一覧は 5.2 章を参照してください.

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

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

3.7. CaoController::get_ExtensionNames プロパティ

Extension 名リストを取得します.

3.8. CaoController::get_VariableNames プロパティ

変数名リストを取得します.

3.9. CaoExtension::get_VariableNames プロパティ

変数名リストを取得します.

3.10. CaoVariable::put_ID プロパティ

CaoVariable の ID を設定します. 設定値は引数で指定した値を設定します.
一部の変数へアクセスする際の ID 指定に用います.

3.11. CaoVariable::get_ID プロパティ

CaoVariable の ID を取得します.

3.12. CaoVariable::put_Value プロパティ

変数名で指定したデータを設定します. 設定値は引数で指定した値を設定します.

3.13. CaoVariable::get_Value プロパティ

変数名で指定したデータを取得します.

4. コマンド一覧

4.1. CaoController クラス

表 4-1 CaoController クラス コマンド一覧

コマンド名	機能	
Raw	汎用コマンドを送出します.	P11

4.1.1. CaoController::Execute("Raw") コマンド

汎用コマンドの送出を行う.

書式 Raw (<vntDataArray>)

<vntDataArray> : [in] コマンドデータ
(VT_UI1 | VT_ARRAY)
[0] サービスコード
[1] クラス ID
[2] インスタンス ID
[3] アトリビュート ID(Low)
[4] アトリビュート ID(High)
[n] サービスデータ

- アトリビュート ID が 8Bit の場合は High 側を 0x00 で指定すること
- 要素 5 以降はサービスデータとして扱う
- サービスデータは Low 側から 8Bit ずつ指定すること
- サービスデータが奇数 Byte の場合は内部で NULL(0x00) パディングされる

戻り値 : [out]応答データ
(VT_UI1 | VT_ARRAY)

- 応答データが無いものは VT_EMPTY が返る
- 応答データは Low 側から 8Bit ずつ格納される

4.2. CaoExtention クラス

表 4-2 CaoExtention クラス コマンド一覧

コマンド名	機能	
Preset	プリセットを行います.	P12
PresetReset	プリセットリセットを行います.	P12
Reset	リセットを行います.	P12
InitReset	イニシャルリセットを行います.	P13
ErrClear	エラークリアを行います.	P13

4.2.1. CaoExtention::Execute(“Preset”) コマンド

プリセット実行要求を行う.

書式 Preset ()

引数 : 無し

戻り値 : 無し

4.2.2. CaoExtention::Execute(“PresetReset”) コマンド

プリセットリセット実行要求を行う.

書式 PresetResett ()

引数 : 無し

戻り値 : 無し

4.2.3. CaoExtention::Execute(“Reset”) コマンド

リセット実行要求を行う.

書式 Reset ()

引数 : 無し

戻り値 : 無し

4.2.4. CaoExtention::Execute(“InitReset”) コマンド

イニシャルリセット実行要求を行う.

書式 InitReset ()

引数 : 無し

戻り値 : 無し

4.2.5. CaoExtention::Execute(“ErrClear”) コマンド

エラークリア実行要求を行う.

書式 ErrClear ()

引数 : 無し

戻り値 : 無し

5. 変数一覧

5.1. CaoController クラス

5.1.1. システム変数

表 5-1 CaoController クラス システム変数一覧

変数名	データ型	説明	属性	
			get	put
@MAKER_NAME	VT_BSTR	メーカ名=”KEYENCE”を返す.	○	―
@VERSION	VT_BSTR	プロバイダバージョン情報を返す.	○	―
@LAST_RESPONSE_STATUS	VT_I4 | VT_ARRAY	直前のエラー応答詳細情報. (VT_I4 | VT_ARRAY) [0] 一般ステータス [1] 追加ステータス(無い場合は 0 セット)	○	―

5.1.2. ユーザ変数

表 5-2 CaoController クラス ユーザ変数一覧

変数名	データ型	説明	属性	
			get	put
無し			―	―

5.2. CaoExtention クラス

5.2.1. システム変数

表 5-3 CaoExtention クラス システム変数一覧

変数名	データ型	説明	属性	
			get	put
@CriteriaValue	VT_R8	判定値(P.V.値) -199.9999～199.9999(初期値:0.0000)	○	―
@CalcValue	VT_R8	演算表示値 -199.9999～199.9999(初期値:0.0000) ※ 演算機能を使用しない場合，読み出しエラーが発生します	○	―
@RawValue	VT_R8	生値(R.V.値) -199.9999～199.9999(初期値:0.0000)	○	―
@PeakValue	VT_R8	P-P モード時のピークホールド値 -199.9999～199.9999(初期値:0.0000) ※ 検出モードがピーク to ピークモード以外の場合，読み出しエラーが発生します	○	―
@BottomValue	VT_R8	P-P モード時のボトムホールド値 -199.9999～199.9999(初期値:0.0000) ※ 検出モードがピーク to ピークモード以外の場合，読み出しエラーが発生します	○	―
@ControlOutput	VT_I4	制御出力 0x00～0x1F(初期値:0x00) bit0 : HIGH 出力 bit1 : LOW 出力 bit2 : GO 出力 bit3 : HH 出力 bit4 : LL 出力	○	―

@ErrStatu	VT_I4	センサアンプエラー状態 0x00～0xFF bit0 ： 過電流エラー（ErC） bit1 ： ヘッドエラー（ErH） bit2 ： EEPROM エラー（ErE） bit3 ： ひっかかりチェックエラー（Er.chK） bit4 ： セルフタイミングディレイ（Er.dLY） bit5 ： ユニット台数エラー（Er.Unit） bit6 ： 演算エラー（Er.CAL） bit7 ： 演算専用モードエラー（Er.noH）	○	－
@CalcRawValue	VT_R8	演算結果の元となった R.V.値 -199.9999～199.9999(初期値:0.0000) ※ 機器 ID は ID プロパティにて指定(ID=1～15) ※ 演算機能を使用しない場合，読み出しエラーが発生します	○	－
@Preset	VT_I4	プリセット実行要求 R 0～2 ： 最後に書き込んだ値 W 0→1 ： プリセット実行 0→2 ： プリセットリセット実行	○	○
@Reset	VT_I4	リセット実行要求 R 0～1 ： 最後に書き込んだ値 W 0→1 ： リセット実行	○	○
@InitReset	VT_I4	イニシャルリセット実行要求 R 0～1 ： 最後に書き込んだ値 W 0→1 ： イニシャルリセット実行	○	○
@ErrClear	VT_I4	エラークリア実行要求 R 0～1 ： 最後に書き込んだ値 W 0→1 ： エラークリア実行	○	○

@Bank	VT_I4	バンク選択状態 0～3(初期値:0)	○	○
@Timing	VT_I4	タイミング状態 0～1(初期値:0) タイミング入力 OFF R 0 : またはセルフタイミングで測定中 タイミング入力 ON 1 : またはセルフタイミングで非測定中 W 0 : 測定中に変更 1 : 非測定中に変更	○	○
@KeyLock	VT_I4	キーロック 0～2(初期値:0) 0 : アンロック 1 : フル・キーロック 2 : キーロック	○	○
@BarDisp	VT_I4	バー表示モード 0～1(初期値:0) 0 : バー表示モード 1 : OK/NG 表示モード	○	○
@Bank_HH	VT_R8	バンク*の HH 側設定値 -199.9999～199.9999(初期値:7.0000) ※ バンク No は ID プロパティにて指定(ID=0～3)	○	○
@Bank_HIGH	VT_R8	バンク*の High 側設定値 -199.9999～199.9999(初期値:5.0000) ※ バンク No は ID プロパティにて指定(ID=0～3)	○	○
@Bank_LOW	VT_R8	バンク*の Low 側設定値 -199.9999～199.9999(初期値:1.0000) ※ バンク No は ID プロパティにて指定(ID=0～3)	○	○

@Bank_LL	VT_R8	バンク*の LL 側設定値 -199.9999～199.9999(初期値:-1.0000) ※ バンク No は ID プロパティにて指定(ID=0～3)	○	○
@Bank_Preset	VT_R8	バンク*のプリセット側設定値 -199.9999～199.9999(初期値:0.0000) ※ バンク No は ID プロパティにて指定(ID=0～3)	○	○
@CalcMode	VT_I4	演算モード・演算設定 0～27(初期値:0) 10 の位（演算モード） 0 : 演算機能を使用しない 1 : 演算機能を使用する 2 : 演算専用モード 1 の位（演算設定） 0 : 最大値 1 : 最小値 2 : 平坦度 3 : 平均値 4 : 基準差 5 : ねじれ 6 : 反り 7 : 厚み	○	○
@DetectingMode	VT_I4	検出モード 0～4(初期値:0) 0 : スタンダード 1 : NG ホールド 2 : ピーク 3 : ボトム 4 : ピーク to ピーク	○	○
@HoldUpdate	VT_I4	ホールド更新方式 0～1(初期値:0) 0 : タイミング 1 : 常時更新	○	○

@ResponseTime	VT_I4	応答時間 0～5(初期値:3) 接続しているセンサヘッドが GT2-H***, A*** の場合 0 : HSP(3.0ms) 1 : 5ms 2 : 10ms 3 : 100ms 4 : 500ms 5 : 1000ms 接続しているセンサヘッドが GT2-P*** の場合 0 : HSP(12ms) 1 : 20ms 2 : 40ms 3 : 400ms 4 : 2000ms 5 : 4000ms	○	○
@TimingCategory	VT_I4	タイミング種別 0～2(初期値:0) 0 : 外部タイミング入力 1 : 立ち上がりセルフ 2 : 立ち下がりセルフ	○	○
@TimingLevel	VT_R8	セルフタイミングレベル -199.9999～199.9999(初期値:0.5000)	○	○
@TimingDelay	VT_I4	セルフタイミングディレイ種別 0～1(初期値:0) 0 : スタティックホールド 1 : ディレイタイマ	○	○
@DelayTime	VT_I4	ユーザ指定ディレイ時間 0～9999(初期値:1000)	○	○
@JudgeDelay	VT_I4	スタティックホールド ディレイ安定判断 0～1(初期値:0) 0 : デフォルト 1 : ユーザ	○	○

@HoldRange	VT_R8	スタティックホールド ディレイ安定幅 0.0000～199.9999(初期値:0.0100)	○	○
@MeasureDirect	VT_I4	計測増減方向 0～1(初期値:0) 0 : 通常 1 : 反転	○	○
@LeverRatio	VT_R8	レバー比 0.1～100.0(初期値:1.0)	○	○
@Output	VT_I4	出力様式 0～1(初期値:0) 0 : N.O. 1 : N.C.	○	○
@DisplayDigit	VT_I4	表示桁数 0～3(初期値:0) 0 : 0.0001 1 : 0.001 2 : 0.01 3 : ___0.1	○	○
@Hysteresis	VT_R8	ヒステリシス 0.0000～199.9999(初期値:0.0030)	○	○
@AllInput	VT_I4	一斉出力設定 0～1(初期値:0) 0 : 個別入力 1 : 一斉入力	○	○
@SpecialOutput	VT_I4	特別出力設定 0～5(初期値:0) 0 : 使用しない 1 : 5 出力 2 : リミット出力 3 : リミット出力ユーザ設定 4 : オール GO 5 : オールリミット出力	○	○
@Limit_HH	VT_R8	リミット出力 HH 側判定位置設定 -199.9999～199.9999(初期値:0.5000)	○	○
@Limit_LL	VT_R8	リミット出力 LL 側判定位置設定 -199.9999～199.9999(初期値:0.5000)	○	○

@PresetData	VT_I4	プリセットデータ選択 0～1(初期値:0) 0 : R.V. 値 1 : P.V. 値	○	○
@PresetMem	VT_I4	プリセット記憶 0～1(初期値:0) 0 : YES 1 : NO	○	○
@PresetPoint	VT_I4	プリセットポイント 0～1(初期値:0) 0 : 全バンク共通 1 : バンクごとに記憶	○	○
@EcoMode	VT_I4	省電力機能(ECO) 0～2(初期値:0) 0 : OFF 1 : HALF 2 : ALL	○	○
@CheckHook	VT_I4	ひっかかりチェック機能 0～2(初期値:0) 0 : OFF 1 : ON 2 : USER	○	○
@CheckHookPos	VT_R8	ひっかかりチェック動作位置 -199.9999～199.9999(初期値:0.5000)	○	○
@BunchSetting	VT_I4	一括設定 0～1(初期値:0) 0 : 個別 1 : 一括 ※ GT2-100 シリーズのみ使用できます	○	○
@AnalogSetting	VT_I4	アナログレンジ設定 0～1(初期値:0) 0 : デフォルト 1 : フリーレンジ設定 ※ GT2-71MC*のみ使用できます	○	○

@FreeRange_Hi	VT_R8	フリーレンジ設定(Hi 側) -199.9999～199.9999(初期値:12.0000) ※ GT2-71MC*のみ使用できます	○	○
@FreeRange_Lo	VT_R8	フリーレンジ設定(Lo 側) -199.9999～199.9999(初期値:0.0000) ※ GT2-71MC*のみ使用できます	○	○
@LAST_RESPONSE_STATUS	VT_I4 | VT_ARRAY	直前のエラー応答詳細情報. (VT_I4 | VT_ARRAY) [0] 一般ステータス [1] 追加ステータス(無い場合は 0 セット)	○	―

5.2.2. ユーザ変数

表 5-4 CaoController クラス ユーザ変数一覧

変数名	データ型	説明	属性	
			get	put
無し			―	―

6. エラーコード

GT2DLEP1 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 6-1 固有エラーコード

エラー名	エラー番号	説明
CIP 応答異常 (フォーマットエラー)	0x80100000	CIP 通信の応答パケットが想定外の異常なフォーマットであった場合に返ります．
CIP 応答異常 (Error Status Code)	0x80100001	CIP 通信の応答パケットがステータス異常であった場合に返ります．
CIP 応答異常 (Error General Status)	0x80100002	CIP 通信の応答パケットがエラー応答であった場合に返ります． 本エラーを受信した場合は，システム変数の @LAST_RESPONSE_STATUS を読み出すことで詳細エラーコードを得ることができます．

表 6-2 詳細エラーコード

一般ステータス	追加ステータス	説明
0x00	0x0000	正常終了．
0x05	0x0000	インスタンス ID が範囲外です．
0x08	0x0000	指定したインスタンス ID ではこのサービスは非対応であるか，実行できません．
0x09	0x0000	書き込まれたデータが範囲外です．
0x0C	0x0000	動作指令の実行に失敗しました．センサアンプが，動作指令が実行できる状態か確認してください．
0x0E	0x0000	・ 書き込みできないアトリビュート ID，または書き込みできない状態のアトリビュート ID に書き込もうとしています． ・ 書き込まれたデータが範囲外です．
0x10	0x0000	読み出しできないアトリビュート ID，または読み出しできない状態のアトリビュート ID から読み出そうとしています．
0x13	0x0000	サービスデータのサイズが規定以下です． 追加ステータスには，規定されたデータサイズが格納されます．

0x14	0x0000	アトリビュート ID が範囲外です.
0x16	0x0000	指定されたインスタンス ID に対応するセンサアンプが接続されていません.
0x1F	0xC350	指定されたインスタンス ID でこのサービスはサポートされていますが, 指定されたアトリビュート ID では使用できません.
	0xC351	現在のモードでは指定した設定値をパラメータに対して書き込みできません.
	0xC352	本機が通信初期化中です.
0xFE	0x0000	システムエラー.

7. サンプルプログラム

以下にサンプルコード(C#)を示します.

7.1. 接続とオブジェクトの生成

```
//  通信ユニットに接続しアンプユニットやセンサヘッドにアクセスするオブジェクトを生成
private void Form1_Load(object sender, EventArgs e)
        {
            try
            {
                // CAO エンジンの生成
                this.eng = new CCaoEngine();

                // 通信ユニットへの接続
                this.ctrl = this.eng.Workspaces[0].AddController(
                                        "Sample",
                                        "CaoProv.KEYENCE.GT2DLEP1",
                                        string.Empty,
                                        "CONN=ETH:192.168.0.20");

                // アンプユニットオブジェクト生成
                this.ext = this.ctrl.AddExtension("CH1", string.Empty);

                // 変数オブジェクト生成
                this.wVal = this.ext.AddVariable("@Bank_HIGH", string.Empty);
                this.rVal = this.ext.AddVariable("@CriteriaValue", string.Empty);
            }
            catch (Exception ex)
            {
                MessageBox.Show(this, ex.Message);
            }
        }
```

7.2. センサ ID=1 のリセット

```
//  センサ ID=1 をリセット
private void Command1_Click(object sender, EventArgs e)
        {
            try
            {
                // リセット要求
                this.ext.Execute("Reset", null);
            }
            catch (Exception ex)
            {
                MessageBox.Show(this, ex.Message);
            }
        }
```

7.3. センサ ID=1 のバンク 2 High 側設定値の設定

```
//  バンク No の設定
private void Command2_Click(object sender, EventArgs e)
        {
            try
            {
                // バンク No の設定
                this.wVal.ID = 2;

                // 値の設定
                double val;
                double.TryParse(this.textBox1.Text, out val);
                this.wVal.Value = val;
            }
            catch (Exception ex)
            {
                MessageBox.Show(this, ex.Message);
            }
        }
```

7.4. センサ ID=1 の判定値(P.V.値)を取得

```
//  センサ ID=1 の判定値(P.V.値)を取得
private void Command3_Click(object sender, EventArgs e)
        {
            try
            {
                // 値の取得
                this.textBox2.Text = ((double)(this.rVal.Value)).ToString();
            }
            catch (Exception ex)
            {
                MessageBox.Show(this, ex.Message);
            }
        }
```

7.5. センサ ID=1 の判定値(P.V.値)の取得 (Raw コマンド)

```
private void Command4_Click(object sender, EventArgs e)
        {
            try
            {
                // センサ ID=1 判定値(P.V.値) 取得
                byte[] param = new byte[5];
                param[0] = 0x0E;    // サービスコード          : パラメータ読み出し
                param[1] = 0x67;    // クラス ID               : DL Object クラス
                param[2] = 0x01;    // インスタンス ID         : センサ ID=1
                param[3] = 0x25;    // アトリビュート ID(Low)  : 判定値(P.V.値) (0x0325)
                param[4] = 0x03;    // アトリビュート ID(High)
                object val = this.ctrl.Execute("Raw", param);
                string str = string.Empty;
                foreach (byte b in (byte[])val)
                {
                    str += " ";
                    str += b.ToString();
```

```
            }
            this.textBox3.Text = str;
        }

        catch (Exception ex)
        {
            MessageBox.Show(this, ex.Message);
        }
    }
```

7.6. 通信ユニットのステータスの取得 (Raw コマンド)

```
    private void Command5_Click(object sender, EventArgs e)
    {
        try
        {
            // 通信ユニット ステータス 取得
            byte[] param = new byte[5];
            param[0] = 0x0E;    // サービスコード          : パラメータ読み出し
            param[1] = 0x67;    // クラス ID               : DL Object クラス
            param[2] = 0x00;    // インスタンス ID         : 通信ユニット=0
            param[3] = 0x64;    // アトリビュート ID(Low)  : ステータス (0x64)
            param[4] = 0x00;    // アトリビュート ID(High)
            object val = this.ctrl.Execute("Raw", param);
            string str = string.Empty;
            foreach (byte b in (byte[])val)
            {
                str += " ";
                str += b.ToString();
            }
            this.textBox4.Text = str;
        }
        catch (Exception ex)
        {
            MessageBox.Show(this, ex.Message);
        }
    }
```