Digital Gauge Counter

DG3000 シリーズ

ダイナミックリンクライブラリ
GaugeC48.dll（DLL）
取扱説明書

このダイナミックリンクライブラリ GaugeC48.dll は８ＣＨから４８ＣＨ用の Digital Gauge Counter DG3000 シリーズ共通のＤＬＬです。

この説明書は GaugeC48.dll を使ったアプリケーションを作成するためのものです。

開発環境は Microsoft Visual Basic 6.0 を前提にしております。

このＤＬＬ GaugeC48.dll は作成したアプリケーションの EXE ファイルのあるフォルダまたは Windows の System フォルダに置いてください。

```
' 内部関数をアクセスするユーザー定義型を記述してください。
Type DW_DATA
    Status As Long          'ステータス
    Data As Long            'データ
End Type

Type DW2_DATA
    Status As Long          'ステータス
    Data(1) As Long         '２データ用
End Type

Type VER_INF
    Status As Long          'ステータス
    Info(59) As Byte        'Version 情報文字列(60 Bytes 固定)
End Type                     'ASCII データがセットされます

Type CMD_STS             'USB 送受信用
    whdl As Long         'Write 用デバイスハンドル
    rhdl As Long         'Read 用デバイスハンドル
    derr As Long         'DLL エラー
        '1=PIPE00 Open NG
        '2=PIPE01 Open NG
        '3=PIPE01 Close NG
        '4=PIPE00 Close NG
        '5=Write 用デバイスハンドル INVALID
        '6=Read 用デバイスハンドル INVALID
        '7=受信用バッファ領域確保 NG
        '8=WriteFile エラー
        '9=WriteFile 書き込みバイト数不一致
        '10=ReadFile エラー
        '11=ReadFile 読み込みバイト数不一致
        '12=ReadFile 最初のデータが送信と不一致
        '13=ReadFile 非同期処理のエラー
        '14=ReadFile タイムアウト
    werr As Long         'Windows エラー
                         'derr<>0 の時の GetLastError()
    completeDeviceName(255) As Byte
                'GlobalUniqueID のデバイス情報シンボリックリンク名
End Type
```

```
'グローバル変数またはパブリック変数で定義をしてください。

 Global USBdev As CMD_STS           'USB 送受信用
    USB デバイスのオープン関数 Open_USB() で USBdev を渡します。
  以後はこの USBdev を使って装置にアクセスして下さい。
```

```
'GaugeC48.dll　内部関数の参照宣言をしてください。

'（１）DLL 内部関数の初期化(最初に必ず実行してください)
Declare Sub Init_USB Lib "GaugeC48.dll" ()

'（２）DLL 内部関数の終了処理(最後に必ず実行してください)
Declare Sub Term_USB Lib "GaugeC48.dll" ()

'（３）USB デバイスのオープン
Declare Function Open_USB Lib "GaugeC48.dll"
                                    (ByRef CmdSts As CMD_STS) As Long

'（４）USB デバイスのクローズ
Declare Function Close_USB Lib "GaugeC48.dll"
                                    (ByRef CmdSts As CMD_STS) As Long

'（５）USB デバイスのバージョン番号取得
Declare Function Read_Version Lib "GaugeC48.dll" (ByRef CmdSts As CMD_STS,
                                      ByRef VerInf As VER_INF) As Long

'（６）使用チャンネルを書き込み
Declare Function Write_Channel Lib "GaugeC48.dll"
          (ByRef CmdSts As CMD_STS, ByRef Dw2Data As DW2_DATA) As Long

'（７）使用チャンネルの読み込み
Declare Function Read_Channel Lib "GaugeC48.dll" (ByRef CmdSts As CMD_STS,
                                    ByRef Dw2Data As DW2_DATA) As Long

'（８）サンプリング時間を書き込み
Declare Function Write_SamplingTime Lib "GaugeC48.dll"
            (ByRef CmdSts As CMD_STS, ByRef DwData As DW_DATA) As Long

'（９）サンプリング時間の読み込み
Declare Function Read_SamplingTime Lib "GaugeC48.dll"
            (ByRef CmdSts As CMD_STS, ByRef DwData As DW_DATA) As Long

'（１０）データ数を読み込み
Declare Function Read_DataCount Lib "GaugeC48.dll"
            (ByRef CmdSts As CMD_STS, ByRef DwData As DW_DATA) As Long
```

```
'（１１）カウンタをリセットする
Declare Function Counter_Reset Lib "GaugeC48.dll"
          (ByRef CmdSts As CMD_STS, ByRef Dw2Data As DW2_DATA) As Long

'（１２）手動トリガ(ソフトから)をかける　外部(手動)トリガの場合有効
Declare Function Manual_Trigger Lib "GaugeC48.dll"
            (ByRef CmdSts As CMD_STS, ByRef DwData As DW_DATA) As Long

'（１３）計測開始／停止
Declare Function Measure_StartStop Lib "GaugeC48.dll"
            (ByRef CmdSts As CMD_STS, ByRef DwData As DW_DATA) As Long

'（１４）コンフィグレーション設定
Declare Function Write_Configuration Lib "GaugeC48.dll"
            (ByRef CmdSts As CMD_STS, ByRef DwData As DW_DATA) As Long

'（１５）計測データを取り込み
Declare Function Read_Data Lib "GaugeC48.dll" (ByRef CmdSts As CMD_STS,
                  ByRef DwData As DW_DATA, ByRef buf As Byte) As Long

'（１６）ボード実装状態の読み込み
Declare Function Read_BoardNum Lib "GaugeC48.dll"
          (ByRef CmdSts As CMD_STS, ByRef DwData As DW_DATA) As Long

'（１７）リアルタイムデータを取り込み
Declare Function Read_Real Lib "GaugeC48.dll" (ByRef CmdSts As CMD_STS,
          ByRef DwData As DW_DATA, ByRef buf As Byte) As Long
```

１．DLL 内部関数の初期化
Init_USB()

動作
DLL 内部関数の初期化を行います。
DLL を使う一番最初に必ず実行してください。

２．DLL 内部関数の終了処理
Term_USB()

動作
DLL 内部関数の終了処理を行います。
DLL を終了する最後に必ず実行してください。

３．USB デバイスのオープン
Open_USB(USBdev)

動作
USB デバイスをオープンします。

パラメータ
USBdev = USB 送受信用ユーザー定義型バッファ(参照渡し)

戻り値
0≠関数が成功(装置が接続されている)
0＝関数が失敗(装置が未接続)
関数が成功すると、ハンドル等の情報が USBdev にセットされる。
以後はこの USBdev を使って装置にアクセスします。

４．USB デバイスのクローズ
Close_USB(USBdev)

動作
アプリケーション終了時、装置の USB デバイスをクローズしてください。

パラメータ
USBdev = オープンされている USB 送受信用ユーザー定義型バッファ(参照渡し)

戻り値
0≠関数が成功
0＝関数が失敗
関数が成功すると、ハンドル等の情報がクリアされる。

５．USB デバイスのバージョン番号取得

Read_Version(USBdev, version)

動作

USB デバイスのバージョン番号を読み込みます。

パラメータ

USBdev = USB 送受信用ユーザー定義型バッファ（参照渡し）
version = ユーザー定義型 VER_INF のバッファ（参照渡し）

戻り値

0≠関数が成功
0＝関数が失敗
関数が成功すると、version.Status に装置側のステータスがセットされる。

■ version.Status が０ならば、装置の Version 情報が
version.Info にセットされています。
version.Info は ASCII 文字です。

■ version.Status が０以外ならば、装置側で実行されなかったことを意味します。

６．使用チャンネルの書き込み

Write_Channel(USBdev, channel)

動作

装置に計測用使用チャンネルを書き込みます。

パラメータ

USBdev = USB 送受信用ユーザー定義型バッファ（参照渡し）
channel = ユーザー定義型 DW2_DATA のバッファ（参照渡し）
channel.Data(0) CH 指定値 CH1 から CH24
CH1=&h000001(bit0) ～ CH24 =&h800000(bit23)
channel.Data(1) CH 指定値 CH25 から CH48
CH25=&h000001(bit0) ～ CH48 =&h800000(bit23)
DLL 共通化のため２４チャンネル以下の装置であっても DW2_DATA のバッファでアクセスしてください。
実装されていない CH に 1 を指定しても意味をもちません。

戻り値

0≠関数が成功
0＝関数が失敗
関数が成功すると、channel.Status に装置側のステータスがセットされる。
■ channel.Status が０ならば、使用チャンネルの書き込みが実行されました。
■ channel.Status が０以外ならば、装置側で実行されなかったことを意味します。

７．使用チャンネルの読み込み

Read_Channel(USBdev, channel)

動作

装置の計測用使用チャンネルを読み込みます。

パラメータ

USBdev = USB 送受信用ユーザー定義型バッファ（参照渡し）
channel = ユーザー定義型 DW2_DATA のバッファ（参照渡し）
　DLL 共通化のため２４チャンネル以下の装置であっても DW2_DATA のバッファでアクセスしてください。

戻り値

0≠関数が成功
0＝関数が失敗
関数が成功すると、channel.Status に装置側のステータスがセットされる。

■ channel.Status が０ならば、装置の計測用使用チャンネルが channel.Data にセットされています。

channel.Data(0) CH 戻り値 CH1 から CH24
　CH1=&h000001(bit0) ～ CH24 =&h800000(bit23)
channel.Data(1) CH 戻り値 CH25 から CH48
　CH25=&h000001(bit0) ～ CH48 =&h800000(bit23)
実装されていない CH のビットは意味をもちません。

■ channel.Status が０以外ならば、装置側で実行されなかったことを意味します。

８．サンプリング時間の書き込み

Write_SamplingTime(USBdev, time)

動作

装置にサンプリング時間を書き込みます。
０を書き込むと装置側の自動(内部)サンプリングが停止します。
手動(外部)トリガーの場合はサンプリング時間に０を書き込んでください。

パラメータ

USBdev = USB送受信用ユーザー定義型バッファ（参照渡し）
time = ユーザー定義型 DW_DATA のバッファ（参照渡し）
time.Data にサンプリング時間を指定します。
指定値は２４ビット以下で単位は[ms]です。

戻り値

0≠関数が成功
0＝関数が失敗
関数が成功すると、time.Status に装置側のステータスがセットされる。
■ time.Status が０ならば、サンプリング時間の書き込みが実行されました。
■ time.Status が０以外ならば、装置側で実行されなかったことを意味します。

９．サンプリング時間の読み込み

Read_SamplingTime(USBdev, time)

動作

装置のサンプリング時間を読み込みます。
サンプリング時間が０の場合は、装置側の自動(内部)サンプリングが停止中です。

パラメータ

USBdev = USB送受信用ユーザー定義型バッファ（参照渡し）
time = ユーザー定義型 DW_DATA のバッファ（参照渡し）

戻り値

0≠関数が成功
0＝関数が失敗
関数が成功すると、time.Status に装置側のステータスがセットされる。
■ time.Status が０ならば、装置のサンプリング時間が time.Data にセットされています。
単位は[ms]です。
■time.Status が０以外ならば、装置側で実行されなかったことを意味します。

１０．データ数を読み込み

Read_DataCount(USBdev, count)

動作

装置の計測されたデータ数を読み込みます。
このデータ数とは１データで４８チャンネル全てを指しています。

パラメータ

USBdev = USB 送受信用ユーザー定義型バッファ (参照渡し)
count = ユーザー定義型 DW_DATA のバッファ (参照渡し)

戻り値

0≠関数が成功
0＝関数が失敗
関数が成功すると、count.Status に装置側のステータスがセットされる。

■ count.Status が０ならば、装置のデータ数が count.Data にセットされます。
装置のバッファは１０個あります。
count.Data が１１以上の場合は、装置側でオーバーフローしたことを表し、
この場合、最新の１０個のデータを読んでください。
また、オーバーフローしないように早めに読む必要があります。
■ count.Status が０以外ならば、装置側で実行されなかったことを意味します。

１１．カウンタリセット

Counter_Reset(USBdev, creset)

動作

装置のカウンタをチャンネル指定してリセットします。
全チャンネルリセットの場合は全てのチャンネルを 1 にセットして下さい。

パラメータ

USBdev = USB 送受信用ユーザー定義型バッファ（参照渡し）
creset = ユーザー定義型 DW2_DATA のバッファ（参照渡し）
creset.Data(0) CH 指定値 CH1 から CH24
CH1=&h000001(bit0) ～ CH24 =&h800000(bit23)
creset.Data(1) CH 指定値 CH25 から CH48
CH25=&h000001(bit0) ～ CH48 =&h800000(bit23)
DLL 共通化のため２４チャンネル以下の装置であっても DW2_DATA のバッファでアクセスしてください。
実装されていない CH に 1 を指定しても意味をもちません。

戻り値

0≠関数が成功
0＝関数が失敗
関数が成功すると、creset.Status に装置側のステータスがセットされる。
■creset.Status が０ならば、カウンタリセットが実行されました。
■ creset.Status が０以外ならば、装置側で実行されなかったことを意味します。

１２．手動トリガー

Manual_Trigger(USBdev, trigger)

動作

装置の手動トリガーをアプリケーションから掛ける場合に使います。
実際の運用では装置側のハードで外部トリガを掛けるのでこの関数は必要ないかと思います。
サンプリングが手動(外部)の場合(サンプリング時間に０が設定されている時)有効です。

パラメータ

USBdev = USB 送受信用ユーザー定義型バッファ (参照渡し)
trigger = ユーザー定義型 DW_DATA のバッファ (参照渡し)
trigger.Data の値は何でもかまいません。

戻り値

0≠関数が成功
0＝関数が失敗
関数が成功すると、trigger.Status に装置側のステータスがセットされる。
■ trigger.Status が０ならば、手動トリガーが実行されました。
■ trigger.Status が０以外ならば、装置側で実行されなかったことを意味します。

１３．計測開始／停止

Measure_StartStop(USBdev, st_st)

動作

装置の計測を開始／停止させます。

パラメータ

USBdev = USB 送受信用ユーザー定義型バッファ（参照渡し）
st_st = ユーザー定義型 DW_DATA のバッファ（参照渡し）
　　st_st.Data 指定値：&h01=開始、&h00=停止

戻り値

0≠関数が成功
0＝関数が失敗
関数が成功すると、st_st.Status に装置側のステータスがセットされる。
■ st_st.Status が０ならば、計測開始／停止が実行されました。
■ st_st.Status が０以外ならば、装置側で実行されなかったことを意味します。

１４．コンフィグレーション設定

Write_Configuration(USBdev, config)

動作

装置にコンフィグレーション設定値を書き込みます。

パラメータ

USBdev = USB送受信用ユーザー定義型バッファ（参照渡し）
config = ユーザー定義型 DW_DATA のバッファ（参照渡し）
　　config.Data 指定値
　　bit0(&h01) = 外部トリガー極性、1：立上がり ，0：立下り
　　bit1(&h02) = 外部トリガーイネーブル、1：有効 ，0：無効
　　bit2,3(&h0C) = 分解能選択値
　　0.5[um]=1.0[um]= &h08 , 10.0[um]= &h0C

戻り値

0≠関数が成功
0＝関数が失敗
関数が成功すると、config.Status に装置側のステータスがセットされる。
■config.Status が０ならば、コンフィグレーション設定が実行されました。
■config.Status が０以外ならば、装置側で実行されなかったことを意味します。

１５．計測データを取り込む

Read_Data(USBdev, data, buf)

動作

装置の計測データを取り込みます。

この関数を実行する前に、（１０）「データ数を読み込み」を行ってデータ数を調べてください。

パラメータ

USBdev = USB 送受信用ユーザー定義型バッファ（参照渡し）

data = ユーザー定義型 DW_DATA のバッファ（参照渡し）

data.Data 指定値：データのバイト数を指定する

バイト数の計算方法

（４８チャンネル×４バイト）×データ個数

データ個数とは１０で読み込んだデータ数のことです。

DLL 共通化のため 8CH の装置であっても４８チャンネル×４バイトの単位で転送が行われます。

buf = データバッファ（参照渡し）

このバッファは 1920 Byte 確保してください。

最大量＝{ (48CH X 4Byte) } X 装置側のバッファ個数(10)

戻り値

0≠関数が成功

0＝関数が失敗

関数が成功すると、data.Status に装置側のステータスがセットされる。

■data.Status が０ならば、buf に計測データが読み込まれています。

buf の配列順序はビッグエンディアンで CH1 からです。

buf(0)=CH1 ステータス　bit0:エラー ，bit1:Ready ，bit2:CH 使用中

buf(1)=CH1 データ High Byte

buf(2)=CH1 データ Mid Byte

buf(3)=CH1 データ Low Byte

buf(4)から CH2

・

buf(192)から２個目のステータス・データ開始

■ data.Status が０以外ならば、装置側で実行されなかったことを意味します。

１６．ボード実装状態の読込み

Read_BoardNum(USBdev, bd)

動作

装置のボード実装状態を読み込みます。

パラメータ

USBdev = USB 送受信用ユーザー定義型バッファ (参照渡し)
bd = ユーザー定義型 DW_DATA のバッファ (参照渡し)

戻り値

0≠関数が成功
0＝関数が失敗
関数が成功すると、bd.Status に装置側のステータスがセットされる。

■bd.Status が０ならば、bd.Data にボード実装状態がセットされています。

bd.Data 戻り値
CH1～CH8 実装=&h01(bit0)
CH9～CH16 実装=&h02(bit1)
CH17～CH24 実装=&h04(bit2)
CH25～CH32 実装=&h08(bit3)
CH33～CH40 実装=&h10(bit4)
CH41～CH48 実装=&h20(bit5)

ボード実装は CH1～CH8 から順に始まります。bit が途中で飛ぶ事は有りません。

■ bd.Status が０以外ならば、装置側で実行されなかったことを意味します。

１７．リアルタイムデータを取り込む

Read_Real(USBdev, data, buf)

動作

装置のリアルタイムデータを取り込みます。

パラメータ

USBdev = USB 送受信用ユーザー定義型バッファ（参照渡し）
data = ユーザー定義型 DW_DATA のバッファ（参照渡し）
　data.Data　の値は何でもかまいません。
buf = データバッファ（参照渡し）
　　このバッファは 192 Byte 確保してください。
　　確保するバイト量 = (48CH X 4Byte)
　DLL 共通化のため 8CH の装置であっても 192 バイトの転送が行われます。

戻り値

0≠関数が成功
0＝関数が失敗
関数が成功すると、data.Status に装置側のステータスがセットされる。
■ data.Status が０ならば、buf にリアルタイムデータが読み込まれています。
　　buf の配列順序はビッグエンディアンで CH1 からです。
　　buf(0)=CH1 ステータス　bit0:エラー ，bit1:Ready ，bit2:CH 使用中
　　buf(1)=CH1 データ High Byte
　　buf(2)=CH1 データ Mid Byte
　　buf(3)=CH1 データ Low Byte
　　buf(4)から CH2
■ data.Status が０以外ならば、装置側で実行されなかったことを意味します。