

データロガーからの瞬時値を文字列形式で受信！ テキスト形式や JSON 形式で受信可能なインターフェースコマンド

概要

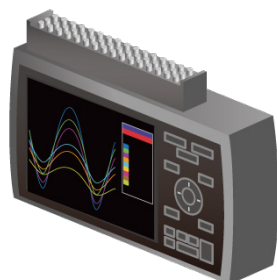
データをバイナリ形式で受信する場合、転送データ容量は少なくなるが、バイナリ形式から物理値 (**V や **°C) への変換が必要となり、プログラムコーディングが多く必要。そこで！

テキスト形式「:MEAS:OUTP:ONECSV?(改行)」
JSON 形式「:MEAS:OUTP:ONEJSON?<改行>」

これらのインターフェースコマンドを使用すると、変換後の数値をそのまま取得可能！

※瞬時値のみ取得可能。 ※機種によって一部コマンドが異なる場合があります。

バイナリ形式のコマンド



GL シリーズ

```
00 13 AA AA 00 01 00 00 02 00 09 11 50 0C D7
0C E1 12 F0 0C F5 02 32 0D 09 0D 13 AA AA 00 01
00 00 00 02 00 09 10 E3 0C D7 0C E1 13 6A 0C F5
01 E8 0D 09 0D 13 AA AA 00 01 00 00 02 00 09
10 89 0C D7 0C E1 13 E4 0C F5 01 3E 0D 09 0D 13
AA AA 00 01 00 00 02 00 09 0F EF E8 F6 E7 00
14 5E E7 14 00 C4 E7 28 E7 32 AA AA 00 01 00 00
00 02 00 09 0F 75 E8 F6 E7 00 14 D8 E7 14 00 4A
E7 28 E7 32 AA AA 00 01 00 00 02 00 09 0E FB
E8 F6 E7 00 15 52 E7 14 FF 00 E7 28 E7 32 AA AA
00 01 00 00 02 00 09 0E 81 E8 F6 E7 00 15 CC
E7 14 FF 56 E7 28 E7 32 AA AA 00 01 00 00 02
00 09 0E 07 E8 F6 E7 00 16 46 E7 14 FE 0C E7 28
E7 32 AA AA 00 01 00 00 02 00 09 0D 8D E8 F6
E7 00 16 C0 E7 14 FE 62 E7 28 E7 32 AA AA 00 01
00 00 02 00 09 0D 13 E8 F6 E7 00 17 3A E7 14
FD E8 E7 28 E7 32 AA AA 00 01 00 00 02 00 09
0C 99 E8 F6 E7 00 17 B4 E7 14 FD 6E E7 28 E7 32
AA AA 00 01 00 00 02 00 09 0C 1F E1 E0 E4 AA
18 2E EA 3D FC F4 EF 01 F2 9B AA AA 00 01 00 00
```

取得したバイナリ形式のデータ

難しい
変換



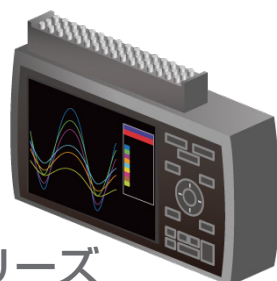
```
+ 48.57
+ 50.00
+1589.7
+ 49.28
+1380.7
+ 36.42
+ 39.99
+1177.0
+ 27.12
+ 20.69
+ 24.26
+ 17.83
```

テキスト形式



画面表示

テキスト形式・JSON 形式のコマンド



GL シリーズ

```
+ 48.57
+ 50.00
+1589.7
+ 49.28
+1380.7
+ 36.42
+ 39.99
+1177.0
+ 27.12
+ 20.69
+ 24.26
+ 17.83
```

取得したテキスト形式データ

とっても
簡単！



画面表示

JSON形式とは

{ } や [] などの括弧を使用して、CHごとに区切る記述方法です。
各プログラミング言語にて、JSONの整形（パース）を行うことができます。

各言語の整形列

JavaScript

```
const value = JSON.parse(jsonData);
```

python

```
value = json.loads(json_data)
```

C#

```
var value = JObject.Parse(jsonData);
```

実際のインターフェースコマンドの応答内容

:MEAS:OUTP:ONECSV?

瞬時値をカンマ区切りで取得できます。文字列の途中で改行はありません。

```
:MEAS:OUTP:ONECSV? ->  
2022/06/16 14:34:41.235,+ 48.57,+ 50.00,+1589.7,+ 49.28,+1380.7,+ 36.42,+ 39.99,+1177.0,  
+ 27.12,+ 20.69,+ 24.26,+ 17.83,+ 11.40,+ 14.97,+ 8.54,+ 2.11,- 4.32,- 0.75,- 7.18,  
- 13.61,LLLLLLLLLLLL,LLLLLLLLLLLL,LLLL
```

:MEAS:OUTP:ONEJSON?

JSON形式で取得できます。文字列の途中で改行はありません。
瞬時値だけでなく、各CHの単位やアラームなど詳細な情報を取得可能です。

※右図は取得したJSON形式の文字列に改行を入れて整形したものです（一部CHのみ）

```
{  
  id: "234"  
  datas: [  
    0: {  
      name: "date"  
      items: {  
        value: "2022/06/16 14:36:01.235"  
      }  
    }  
    1: {  
      name: "ch"  
      items: {  
        ch: "1"  
        value: "-26.60"  
        unit: "V"  
        annot: "CH1"  
        alarm: "L"  
        color: "#E02830"  
      }  
    }  
  ]  
}
```

```
2: {  
  name: "ch"  
  items: {  
    ch: "2"  
    value: "-26.55"  
    unit: "V"  
    annot: "CH2"  
    alarm: "L"  
    color: "#0068B8"  
  }  
}
```

対象機種

コマンド	GL100	GL240	GL840	GLT400	GL260	GL860	GL980 GL2000	GL7000 Plus
:MEAS:OUTP:ONECSV?	-	○ V1.54以降	○ V1.60以降	○ V1.01以降	○ V1.01以降	○ V1.02以降	-	○ V2.00以降
:MEAS:OUTP:DATA:ONECSV?	-	-	-	-	-	-	○ V1.02以降	-
:MEAS:OUTP:ONEJSON?	-	○ V1.54以降	○ V1.60以降	○ V1.01以降	○ V1.01以降	○ V1.02以降	-	○ V2.00以降

コード例 (.NET Framework C# の場合)

本体と USB 接続し、1 秒ごとに本体データを JSON 形式で取得する例です。

※SDK に含まれるサンプルプログラムを使用しています。

```
// USB接続のパラメータ設定
Param param = new Param();
param.UsbId.usbId = 0; // USB ID番号

// 本体に接続
var devIo = new GtcDevIo.DevIo();
var flag = devIo.Open(param);
if (flg == false) return;

for (var i = 0; i < 100; i++)
{
    var json = devIo.SendQuery(":MEAS:OUTP:ONEJSON?");
    var value = JObject.Parse(json);
    Thread.Sleep(1000); // 1秒待つ
}
```

詳細資料をご希望の場合

弊社ホームページよりソフトウェア開発キット (SDK) の申請をお願いいたします。(無償提供)

ホームページ URL



<https://graphteciwatsu.com/jp/support/sdk/>

