

PC 計測 Python モジュール

LabdaqPyModule マニュアル

1、概略

本 PC 計測 Python モジュールは、タートル工業社の以下 A/D ユニットに対応しており、P C に接続された、これら A/D ユニットの計測実行、取り込み等機能を提供するモジュールです。

TUSB-1612ADSM-S2Z	12bit 16ch	0.1V \sim \pm 10V	max100Ks/秒
TUSB-0412ADSM-S2Z	12bit 4ch	0.1V \sim \pm 10V	max100Ks/秒
TUSB-0212ADM2Z	12bit 2ch	0-2V, \pm 1V	max50Ms/秒
TUSB-0216ADMZ	16bit 2ch	\pm 1.25V \sim 10V	max 100Ks/秒
TUSB-0216ADMH	16bit 2ch	0-2V, \pm 1V	max 25Ms/秒
TUSB-K02ADVZ	12bit 2ch	\pm 2.5V	max20Ks/秒

モジュールはモジュール初期化、デバイスセレクト、オープン、計測条件設定、計測開始、ステータス読み込み等、計測に必要な機能を提供しています。

計測は、モジュール内の別スレッドで高速に実行されます、Python 側はステータスを監視しながら、計測終了を待ち、完了後データを受け取ります。また計測中であっても、取得データを受け取ることができます。

動作環境は

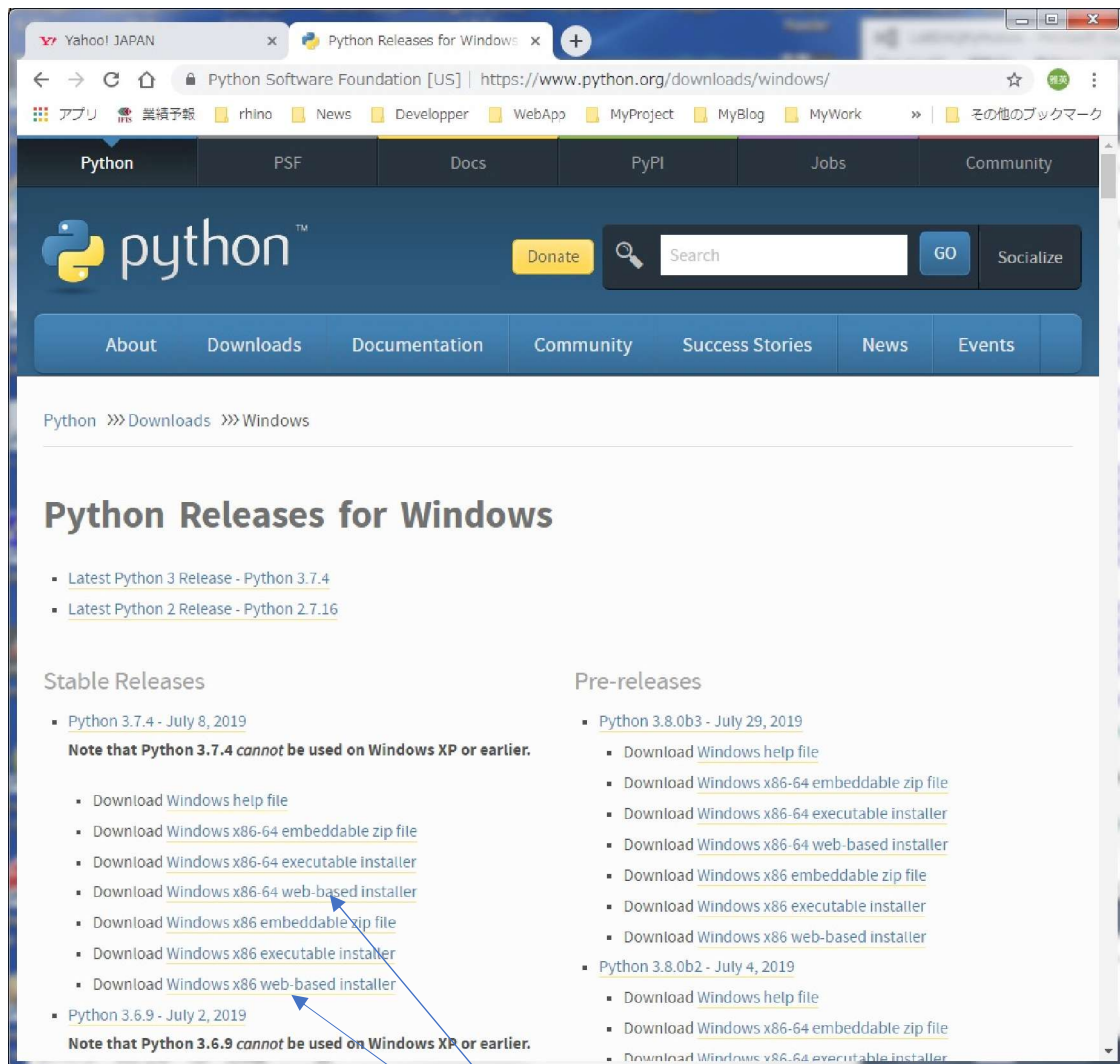
Windows7,8,10 32/64bit .Net Framework 4.5

2、インストール、および環境設定

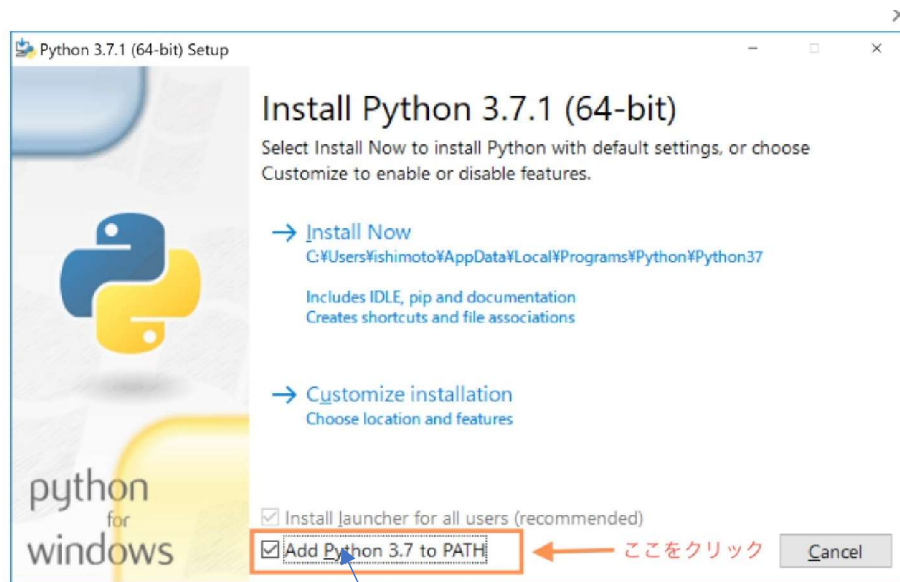
Windows P Cへの Python インストール

以下のサイトからダウンロード インストールができます。

<https://www.python.org/downloads/windows/>

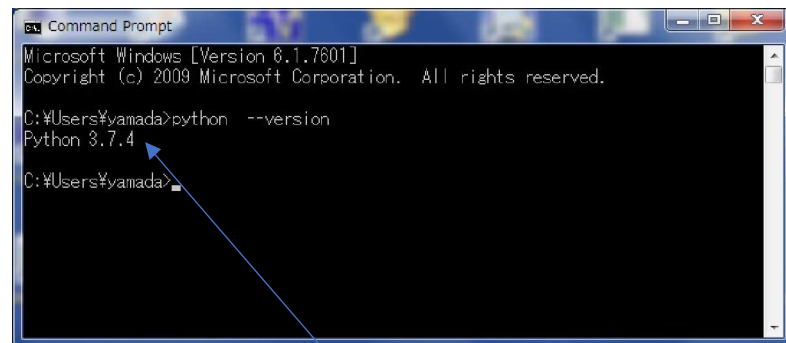


32bit、また 64bit、インストール対象 PC の OS にあわせてダウンロード、インストール実行します。



ここを必ず、チェックオンにし、インストール実行をします。

完了後、コマンドプロンプトで、インストール、およびバージョンを確認します。

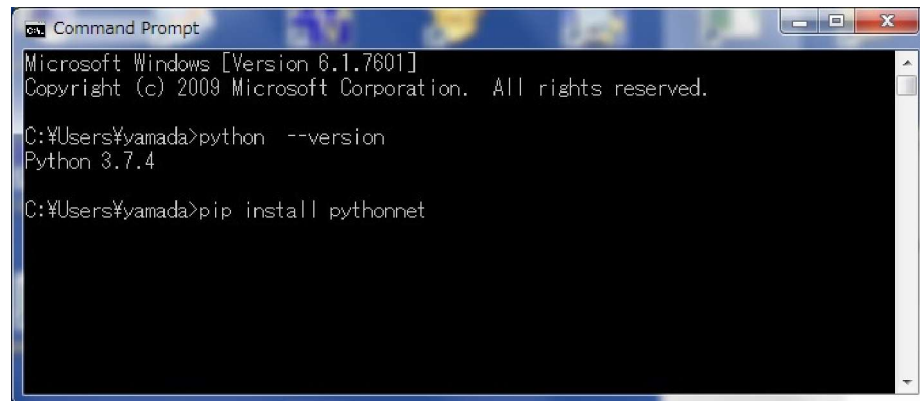


バージョンが表示される

Windows P Cへ Python ドットネットのインストール

本モジュールは C#で開発されており、これを Python から呼び出し可能とするため、Pythonnet(Python for .net - GitHub)のインストールが必要です。

コマンドプロンプトから pip コマンドでインストールします。



```
Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\yamada>python --version
Python 3.7.4

C:\Users\yamada>pip install pythonnet
```

3、PyModule 関数の呼び出し

以下のように clr をインポートして、呼び出します

```
import clr

clr.AddReference('LaBDAQPyModule')
from LaBDAQPyModule import LabdaqTl
```

PyModule の呼び出し手順は

インスタンスの作成

```
labTL = LabdaqTl()
```

モジュールの初期化

```
s = labTL.InitialLabdaqTl()
```

デバイスの選択

```
Id = 0
deviceType = BT_1612ADSM
s = labTL.SelectDevice(deviceType, Id)
```

デバイスのオープン

```
bret = labTL.OpenDevice()
```

計測条件の設定

```
channelNum = input('Enter sampling channel number >')
sampleNum = input('Enter sampling data number >')
bret = labTL.SetSamplingChannelDataNum(channelNum, sampleNum)
```

サンプリング条件のセットアップ

```
bret = labTL.SetupSampling()
```

サンプリング開始

```
bret = labTL.StartSampling()
```

サンプリング終了待ち

```
while True:
```

```
    ステータスの取得
```

```
    status = labTL.GetSamplingStatus()
```

```
    現在までのデータ数の取得
```

```
    count = labTL.GetSamplingDataCount()
```

```
    if status == STS_SAMPEND:
```

```
        s = '計測、正常終了'
```

```
        print(s)
```

```
        count = labTL.GetSamplingDataCount()
```

```
        print('sampling data count = ', count)
```

```
        break
```

終了後、全計測データの取得

```
samplingDataBuff = []
```

```
samplingDataBuff = labTL.GetSamplingData(samplingDataBuff)
```

デバイスのクローズ

```
labTL.CloseDevice()
```

モジュールの開放

```
labTL.TerminateLabdaqTl()
```

上記が基本の手順です、特に、デバイスのクローズ、モジュールの開放の終了処理をしないで、プログラムを終了すると次回、起動しない場合があります。

この場合、一度、A/DユニットをPCから切り離してください。

終了処理は必須です。

4、PyModule 関数一覧

定義定数

#const define

AD ユニット品番

BT_1612ADSM = 0

BT_0412ADSM = 1

BT_0212ADM2 = 2

BT_0216ADMZ = 3

BT_0216ADMH = 4

BT_02ADZ = 5

計測モード

BSMODE_NORMAL = 0

BSMODE_TRIG_REPEAT = 1

計測開始モード

STMODE_IMMEDIATE = 0

STMODE_HDTRIG = 1

サンプリングクロックタイプ

CLOCKMODE_INTERNAL = 0

CLOCKMODE_EXTERNAL = 1

CH2 クロックフェイズ

CLOCKPH_NORMAL = 0

CLOCKPH_REVERSE = 1

デバイスステータス

STS_IDLE = 0

STS_TRGWAIT = 1

STS_TRGON = 2

STS_SAMPEND = 3

STS_SAMPSTOP = 4

STS_SAMPERROR = 5

モジュールの初期化

```
s = labTL.InitialLabdaqTl()
```

AD ユニットの選択

deviceType:

BT_1612ADSM = 0

BT_0412ADSM = 1

BT_0212ADM2 = 2

BT_0216ADMZ = 3

BT_0216ADMH = 4

BT_02ADZ = 5

Id:

```
s = labTL.SelectDevice(deviceType, Id)
```

現在選択の AD ユニットオープン

```
bret = labTL.OpenDevice()
```

現在の設定計測条件の取得

設定計測項目は LaBDAQ5-TL の画面に準拠しています。以下 LaBDAQ5-TL の設定画面を参考にして下さい。

計測実行条件設定

現在のデバイス型番: TUSB-1612ADSM-SZ

開始モード: 即計測

計測チャンネル数: 16

サンプリング点数: 1000

トリガ/ポストトリガ点数: 0

データバッファ点数: 1000

データ取込み点数: 10

入力電圧レンジ: ±10V

サンプリングクロック選択: 内部クロック

サンプリングクロック値(分周比): 1000

サンプリング周期: 1000.000000 us

サンプリング周波数: 1.000000 kHz

ハードウェアトリガ条件

トリガタイプ: 外部デジタル入力立ち上り

トリガチャンネル: チャンネル01

トリガレベル: 1

開始モードがハードウェアトリガ時のみ有効

トリガレベル: -9.9951 Volt (1-4095)

OK キャンセル

計測実行条件設定

現在のデバイス型番	TUSB-0216ADMH		
計測形式	通常計測	トリガ繰り返し計測データ点数	1000
(1-1048576)、かつサンプリング点数以下			
開始モード	即計測	使用/空きメモリ容量	16,000/1,274,392,576 (使用率: 0.001%)
計測チャンネル数	2	計測時間	サンプリング点数 × サンプリング周期 = 0.000040s
サンプリング点数	1000		
プリトリガ/ポストトリガ点数	0	/	1000
プリトリガ + ポストトリガ = サンプリング点数 (ハードウェアプリトリガ時、最大1048576)			
データ取込み点数	10	<= サンプリング点数(計測でデータロスエラー発生する場合、大きくなります)	

入力電圧レンジ	±1V	サンプリングクロック選択	200MHz
		サンプリングクロック周期分周比	8 (8-200)
		平均化回数	1
		サンプリング周期	0.040000usec usec
		サンプリング周波数	25.000000MHz MHz
サンプリング周波数=クロック選択周波数/(周期分周比×平均化回数)、Min1MHz - Max25MHz			

ハードウェアトリガ条件

トリガ発生源	外部トリガ入力(TTLレベル)	開始モードがハードウェアプリトリガ時のみ有効
CH1トリガレベル	1 -1.0000 Volt (1-65534)	
ノイズ除去レベル	0 (0-3277)	

OK キャンセル

計測チャンネル数、計測データ数の取得

channelNum = 0

sampleNum = 0

```
ret, channelNum, sampleNum = labTL.GetSamplingChannelDataNum(
    channelNum,
    sampleNum)
```

基本サンプリング条件の取得

baseMode = 0

baseModeTrigDataNum = 0

startMode = 0

dataBufferSize = 0

dataReadSize = 0

inputRange = 0

```
ret, baseMode, baseModeTrigDataNum, startMode, dataBufferSize, dataReadSize,
inputRange = labTL.GetSamplingMode(baseMode,
    baseModeTrigDataNum,
    startMode,
    dataBufferSize,
    dataReadSize, inputRange)
```

サンプリングクロック条件の取得

samplingClockMode = 0

samplingClockSource = 0

samplingClockDivider = 0

ch2ClockPhase = 0

averageNum = 0

samplingRate, samplingClockMode, samplingClockSource, samplingClockDivider,

ch2ClockPhase, averageNum = labTL.GetSamplingClock(
samplingClockMode,
samplingClockSource,
samplingClockDivider,
ch2ClockPhase, averageNum)

トリガ条件の取得

triggerSource = 0

triggerChannel = 0

triggerLevel = 0

triggerHyLevel = 0

preTriggerNum = 0

ret, triggerSource, triggerChannel, triggerLevel, triggerHyLevel,
preTriggerNum = labTL.GetSamplingTrigger(triggerSource,
triggerChannel,
triggerLevel,
triggerHyLevel,
preTriggerNum)

現在選択 AD ユニットの入力レンジ、クロックソース、トリガソースの一覧
の取得

list = []

list = labTL.GetInputRangeList(list)

list = []

list = labTL.GetClockSourceList(list)

list = []

list = labTL.GetTrigSourceList(list)

計測条件の設定

計測チャンネル数、計測データ数の設定

```
channelNum = input('Enter sampling channel number >')
sampleNum = input('Enter sampling data number >')
ret = labTL.SetSamplingChannelDataNum(channelNum, sampleNum)
```

基本サンプリング条件の設定

```
baseMode = input('Enter base mode(0:normal 1:trigger repeat) >')
baseModeTrigDataNum = input('Enter trigger data number if base mode is trigger
                             repeat >')
startMode = input('Enter start mode(0:immediate 1:hardware trigger) >')
dataBufferSize = input('Enter data buffer size >')
dataReadSize = input('Enter data read size >')
inputRange = input('Enter input range >')
ret = labTL.SetSamplingMode(baseMode,
                             baseModeTrigDataNum,
                             startMode,
                             dataBufferSize,
                             dataReadSize,
                             inputRange)
```

サンプリングクロック条件の設定

```
samplingClockMode = input('Enter clock mode(0:internal 1:external) >')
samplingClockSource = input('Enter clock source >')
samplingClockDivider = input('Enter clock divider >')
ch2ClockPhase = input('Enter ch2 clock phase(0:normal 1:reverse) >')
averageNum = input('Enter averaging number >')
settingClockFreq = labTL.SetSamplingClock(samplingClockMode,
                                             samplingClockSource,
                                             samplingClockDivider,
                                             ch2ClockPhase,
                                             averageNum)
```

トリガ条件の取得

```
triggerSource = input('Enter trigger source >')
triggerChannel = input('Enter trigger channel >')
triggerLevel = input('Enter trigger level >')
triggerHyLevel = input('Enter trigger hysteresis level>')
preTriggerNum = input('Enter pretrigger number >')
bret = labTL.SetSamplingTrigger(triggerSource,
                                triggerChannel,
                                triggerLevel,triggerHyLevel,
                                preTriggerNum)
```

サンプリング条件のセットアップ

```
bret = labTL.SetupSampling()
```

サンプリング開始

```
bret = labTL.StartSampling()
```

サンプリングのステータス取得

```
status = labTL.GetSamplingStatus()
    status == STS_IDLE      '待機中'
    status == STS_TRGWAIT   'トリガ待ち...'
    status == STS_TRGON     '計測中...'
    status == STS_SAMPEND   '計測、正常終了'
    status == STS_SAMPSTOP  '計測が中断されました'
s   tatus == STS_SAMPERROR '計測エラーで中断'
```

現在までのサンプリング点数の取得

```
count = labTL.GetSamplingDataCount()
```

サンプリングエラーコードの取得

```
ret = labTL.GetSamplingErrorCode()
```

サンプリングエラーメッセージの取得

```
ret = labTL.GetSamplingErrorMessage()
```

現在までのサンプリングデータ取得

```
samplingDataBuff = []
```

```
samplingDataBuff = labTL.GetSamplingData(samplingDataBuff)
```

データチャンネル毎に計測データ点数分保存されている配列です

配列 0 ch0 データ (1 回目)

|

配列 N-1 chN-1 データ (1 回目)

配列 N ch0 データ (2 回目)

|

配列 2N-1 chN-1 データ (2 回目)

現在のデバイスのクローズ

```
labTL.CloseDevice()
```

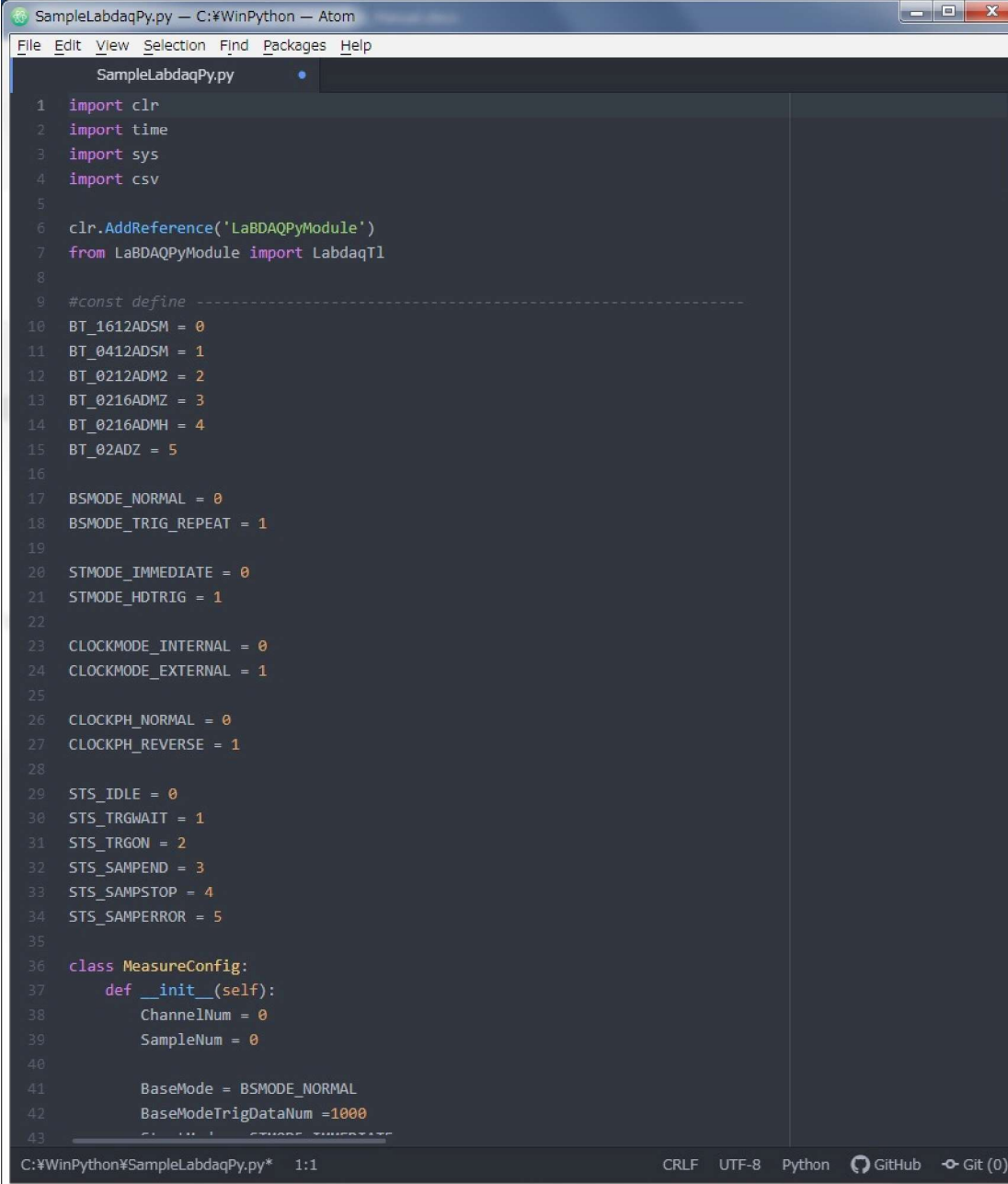
モジュールの開放終了処理

```
labTL.TerminateLabdaqTl()
```

5、サンプルプログラム

pySample フォルダに本 PC 計測モジュールをすべて使用したサンプルが含まれています。

ファイル名は、sampleLabdaqPy.py で、そのまま実行できます。あとモジュール Dll は 32bitOS,64bitOS で異なりますが、パイソンサンプルは同じです。



```
SampleLabdaqPy.py
1 import clr
2 import time
3 import sys
4 import csv
5
6 clr.AddReference('LaBDAQPyModule')
7 from LaBDAQPyModule import LabdaqT1
8
9 #const define -----
10 BT_1612ADSM = 0
11 BT_0412ADSM = 1
12 BT_0212ADM2 = 2
13 BT_0216ADMZ = 3
14 BT_0216ADMH = 4
15 BT_02ADZ = 5
16
17 BSMODE_NORMAL = 0
18 BSMODE_TRIG_REPEAT = 1
19
20 STMODE_IMMEDIATE = 0
21 STMODE_HDTRIG = 1
22
23 CLOCKMODE_INTERNAL = 0
24 CLOCKMODE_EXTERNAL = 1
25
26 CLOCKPH_NORMAL = 0
27 CLOCKPH_REVERSE = 1
28
29 STS_IDLE = 0
30 STS_TRGWAIT = 1
31 STS_TRGON = 2
32 STS_SAMPEND = 3
33 STS_SAMPSTOP = 4
34 STS_SAMPERROR = 5
35
36 class MeasureConfig:
37     def __init__(self):
38         ChannelNum = 0
39         SampleNum = 0
40
41         BaseMode = BSMODE_NORMAL
42         BaseModeTrigDataNum = 1000
43         STS_MODE = STS_IDLE
```

計測テスト、グラフチャートソフトウェア開発

eLaBNET

お問い合わせは089-957-2243

info@labnet.co.jp

株式会社 松山アドバンス 愛媛県松山市古川西 2 丁目 11-24

TEL 089-957-2243

FAX 089-958-2143

www.elabnet.jp