Pythonでパルスジェネレータを制御する方法（2026年版）

～ SCPIコマンドによる外部制御入門 ～

はじめに

Quantum Composers製パルスジェネレータは、SCPI（Standard Commands for Programmable Instruments）コマンドによる外部制御に対応しています。

Pythonを利用することで、

- パルス条件の自動変更
- 条件スイープ
- 長時間評価試験
- 他の計測器との連携

などを実現できます。

本記事では、Python環境の準備から、パルスジェネレータとの通信確認、基本的なSCPI制御までを紹介します。

※「なぜ実験自動化を行うのか」「どのような応用が可能か」については、別資料「AI時代の実験自動化」をご参照ください。

SCPIコマンドとは

SCPI（Standard Commands for Programmable Instruments）は、計測器制御のための標準コマンド体系です。

Quantum Composers製パルスジェネレータでは、テキスト形式のコマンドを送信することで各種設定を変更できます。

例：

```
:PULSE1:DELAY 1E-6
:PULSE1:WIDTH 100E-9
```

この例では、

- Channel 1 Delay = 1 μs
- Channel 1 Width = 100 ns

を設定しています。

【ヒント】

GUIソフトウェアのSCPIログ機能を利用すると、GUI操作時に送信されるSCPIコマンドを確認できます。
設定変更時に実際にどのSCPIコマンドが送信されているかを確認できるため、Python制御用のSCPIコマンド作成にも役立ちます。

---

1. 動作環境の準備

1.1 使用するソフトウェア

本記事では、以下のソフトウェアを使用します。

ソフトウェア	役割
VS Code	Pythonプログラムを編集・実行するための統合開発環境（エディター）
Python 3.x	プログラムを実行するための言語・実行環境
Python拡張機能（Microsoft製）	VS CodeでPythonを編集・実行するための機能
pySerial	PythonからUSB（シリアル通信）を利用できるようにする機能

Pythonが実際にプログラムを実行します。

VS Codeはプログラムを書くためのエディターです。

本記事では、プログラムの編集や実行がしやすい VS Code を使用します。

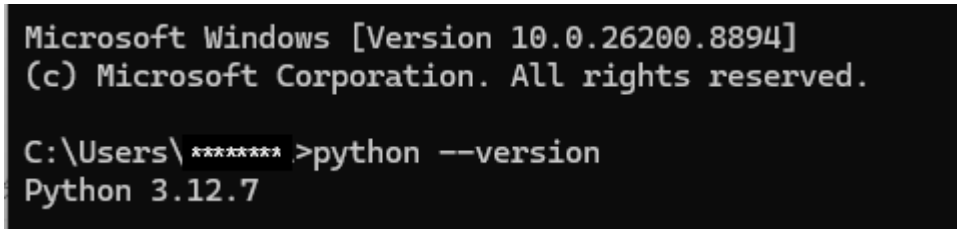
また、VS CodeおよびPython拡張機能が導入済みであることを前提とします。導入方法については、「VS Code導入ガイド」をご参照ください。

IDLEなどのPythonエディターでも同様に作成・実行できますので、ご使用の環境に合わせてお選びください

1.2 準備手順

1. **VS Codeをインストール**
2. **Python拡張機能（Microsoft製）をインストール**
3. **Python 3.xをインストール**
4. **コマンドプロンプトでPythonが動作することを確認**

   python --version



5. **pySerialをインストール**

   python -m pip install pyserial

```
C:\Users\ ******** >python -m pip install pyserial
```

6. **pySerialが利用できることを確認**

   import serial

   print(serial.__version__)

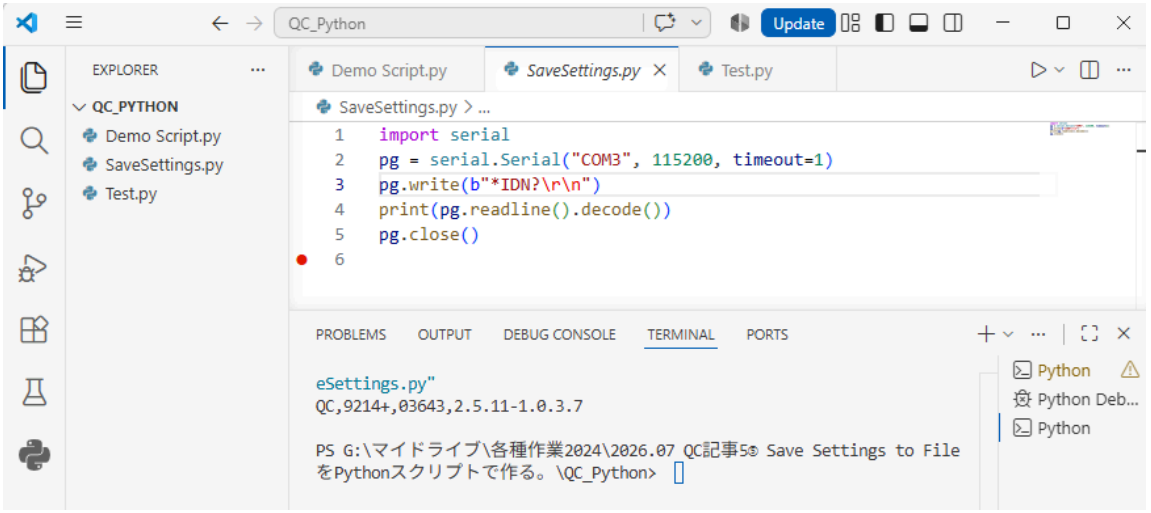
```
C:\Users\ ******** >py
Python 3.12.7 (tags/v3.12.7:0b05ead, Oct  1 2024, 03:06:41) [MSC v.1941 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import serial
>>> print(serial.__version__)
3.5
>>> exit()

C:\Users\ ******** >
```

7. **COMポート番号を確認**

2. パルスジェネレータとの通信確認

VS Codeで実際にPythonプログラムを実行してみましょう。

まずは、SCPI共通コマンド **`*IDN?`** を送信し、パルスジェネレータから機器情報が返ることを確認します。



3. まとめ

本記事では、Python環境の準備から、Quantum Composers製パルスジェネレータとの通信確認までを紹介しました。

まずは、SCPI共通コマンド *IDN? を送信し、機器情報が正常に取得できることを確認してください。通信確認ができれば、PythonからDelay、Width、Periodなどの設定変更や、自動測定・条件スイープなどへ発展させることができます。

本記事が、Pythonによるパルスジェネレータ制御を始める際の一助となれば幸いです。

Appendix　サンプルプログラム

Pythonでカメラシャッターのタイミングをプログラム制御

以下のサンプルは、PythonからSCPIコマンドを送信し、Channel BのDelayを5 μs刻みで変更する例です。レーザー照射に対するカメラシャッターのタイミング変更などをイメージしたサンプルです。

<<サンプル>>

```
import serial
import time

# ----------------------------
# 通信設定
# ----------------------------
PORT = "COM3"
BAUD = 115200

pg = serial.Serial(PORT, BAUD, timeout=1)
def send(cmd):

    """SCPIコマンド送信"""
    pg.write((cmd + "\r\n").encode("ascii"))
    print(cmd)

try:
    # ----------------------------
    # 工場出荷設定
    # ----------------------------
    send("*RST")
    time.sleep(1)

    # ----------------------------
    # System設定
    # ----------------------------
    send(":PULSE0:PERIOD 0.0001")      # 100 μs

    # ----------------------------
    # Channel A（固定）
    # ----------------------------
    send(":PULSE1:DELAY 0.0")
    send(":PULSE1:WIDTH 0.000003")     # 3 μs

    # ----------------------------
    # Channel B
    # ----------------------------
    send(":PULSE2:WIDTH 0.000003")     # 3 μs

    # ----------------------------
    # RUN
    # ----------------------------
    send(":PULSE0:STATE ON")
    print("Delay Sweep Start")

    # Delayを0〜45 μsまで5 μs刻みで変更
    for i in range(10):

        delay = i * 0.000005            # 5 μs
        send(f":PULSE2:DELAY {delay:.6f}")
        print(f"ChB Delay = {delay*1e6:.0f} μs")
        time.sleep(2)

finally:
    # STOP
    send(":PULSE0:STATE OFF")
    pg.close()

print("Finished")
```