```
# _*_ coding: utf-8 _*_
#
import  spidev
import  time

class Adt7310:
    def __init__(self):     #インスタンス作成時に呼び出し 確認2018/12/29
        self.spi = spidev.SpiDev()
        self.spi.open(0,0)
        self.spi.max_speed_hz = 10  # 通信速度の設定
        self.spi.bits_per_word = 8  # 送信文字(bit)単位の設定
        self.spi.mode = 3          #  [CPOL|CPHA]
#  CPOL 0:正理論  ポジパルス ___-_-_-__
#0      CPHA 0: CPOL立上りで読込
#1      CPHA 1: CPOL立下りで読込
#   CPOL 1:負理論 ネガパルス ---_-_-_--
#2      CPHA 0: CPOL立下りで読込
#3      CPHA 1: CPOL立上りで読込
# 参考資料 http://dlnware.com/theory/SPI-Transfer-Modes
#
    def spi_close(self):    #動作未確認
        self.spi.close()

#シャットダウン
#ADT7310は、コンフィギュレーション・レジスタ（レジスタ・アドレス0x01）のビット[6：5]を11にする。
#シャットダウン前の最後の変換からの変換結果は、シャットダウン・モードのときでもADT7310から読み取ることができます。
#コンフィギュレーション・レジスタ（レジスタ・アドレス0x01）のビット[6：5]に00にするとシャットダウン・モードから抜けることができます。
#シャットダウン・モードから出るのに1ms（0.1μFデカップリング・コンデンサで）必要です。
#デバイスがシャットダウンモードから抜け出ると、内部クロックが開始され、変換が開始されます。

    def spi_shutdown(self): #動作未確認
        self.spi.xfer([0b00001000,0b01100000])   # Register:01へ  シャットダウンコマンドdata:0x60 の書き込み
#
#  xfer     CS端子が転送中は  activate  され  未使用時は解放される。
#  xfer2    CS端子は解放されている。
# 参考資料   https://pypi.org/project/spidev/
#
#ワンショットモード
#ワンショット・モードでは、ADT7310はただちに変換を完了してからシャットダウン・モードなる。
```

#ワンショットモードは、消費電力を削減する。
#ワンショットモードを有効にするには、
#コンフィギュレーションレジスタ（レジスタアドレス0x01）のビット[6：5]を01に設定します。
#動作モードビットに書き込んだ後、温度値レジスタから温度を読み出す前に、少なくとも240ms待機が必要。
#この遅延により、ADT7310は起動して変換を完了するのに十分な時間を確保できる。
#更新された温度変換を得るには、
#コンフィギュレーションレジスタ（レジスタアドレス0x01）のビット[6：5]を01にリセットします。
#1 SPSモード
#このモードでは、パーツは1秒あたり1回の測定を実行します。
#一回の変換はわずか60msしかかからず、残りの940msの間はアイドル状態のままです。
#このモードは、コンフィギュレーションレジスタ（レジスタアドレス0x01）のビット[6：5]を10にする。

```
    def spi_oneshot(self):  #動作未確認
        self.spi.xfer([0b00001000,0b10100000])   # Register:01へ　ワンショットコマンドdata:0xa0 の書き込み

    def spi_SPS(self):      #確認2018/12/29
        self.spi.xfer([0b00001000,0b01000000])   # Register:01へ SPSコマンドdata:0x40 の書き込み

#温度データの読込
#読取りは、ADT7310に書込み/読込みビットを1にセットして開始すると、
#アドレス指定されたレジスタに応じて8または16クロック・パルスを供給し、
#ADT7310はアドレス指定されたレジスタからデータをクロック・アウトしますDOUTに出力される。
#データは、コマンドバイトに続くSCLKの最初の立ち下がりエッジでクロック出力されます。
#マスターがCSをハイにすると、読み出しは終了します。
#マスタは、読み込まれたレジスタごとにバス上で新しい読み取りを開始する必要があります。
# 1つの読み取りにつき　1つのレジスタのみが読み出されます。
# ただし、連続リードモードでは、コマンドバイトC2 = 1となり、連続して温度値レジスタを読み出すことができます。
# マスタはSCLKで16クロック・パルスを送信し、温度値はDOUTでクロック出力されます。

    def spi_read(self):     #確認2018/12/29
        e = self.spi.xfer([0b01010000,0xff,0xff])[1:]    # Register:02へ　データ呼び出しコマンドdata:0x50 の書き込み
        er = e[0]<<8 | e[1]
```

```
        if(er >= 0x8000):
            er = er - 0x10000
        er = er / 128.0
        return er                           # ２バイトの0xffはダミーデータです。

#Status Register (Register Address 0x00)
#Bit     Default Type    Name
#[3:0]   0000    R       unused
#4       0       R       Tlow
#5       0       R       Thigh
#6       0       R       Tcrit
#7       0       R       _RDY

#4このビットは、温度がTLOW温度制限を下回ると1に設定されます。
#     ステータスレジスタが読み出されたとき、
#     および/または測定された温度が限界を超えたときに、
#     ビットは0にクリアされます
#     TLOW + THYST設定値レジスタに設定します。
#5このビットは、温度がTHIGHの温度制限を超えると1に設定されます。
#     ステータスレジスタが読み出されたとき、
#     および/または測定された温度が制限値を下回ったとき、
#     ビットは0にクリアされます
#     THIGH–THYST設定値レジスタに設定します。
#6このビットは、温度がTCRITの温度制限を超えると1に設定されます。
#     ステータスレジスタが読み出されたとき、
#     および/または測定された温度が制限値を下回ったときに
#     0にクリアされます
#     TCRIT  –  THYST設定値レジスタに設定します。
#7温度変換結果が温度値レジスタに書き込まれると、このビットはローになります。
#     温度値レジスタを読み出すと1にリセットされます。
#     ワンショットモードと1つのSPSモードでは、
#     このビットはワンショット・ビットへのライト後にリセットされます。

    def spi_status(self):   #確認2018/12/29
        d = self.spi.xfer([0b01000000,0xff])     ## Register:00へ　ステータス呼び出しとダミーdata:0xff の書き込み
        return d[1]

#------------------------ サブルーチン ------------------------
# one SPSモードでは、１秒間に１回の測定が行われるから、測定の終了を待って読み出す部分が必要です。

    def spi_read_SPS(self): #確認2018/12/29
        while self.spi_status() & 0x80 != 0: #  データの準備ができるまで
            time.sleep(0.4)                  #  間隔で待つ。
```

```
        ret = self.spi_read()                   # 温度データを読み出す。
        return ret
#
#
#     ADT7310 の概要
#     ●高性能：
#     温度精度±0.5℃(@−40℃～+105℃（2.7V～3.6V） ±0.4℃(@−40℃～+105℃（3.0V）
#     16ビット温度分解能：0.0078℃
#     高速な最初の温度の変換時間：パワアップ後6ms
#     ●低消費：
#     パワーセービング・モード：1サンプル/1秒
#     ノーマル・モード：700μW@3.3V
#     シャットダウン・モード：7μW@3.3V
#     ●容易な導入：
#     温度校正 / 補正は不要
#     直線性補正の不要
#     ●広い動作範囲：
#     温度範囲：−55℃～+150℃
#     電圧範囲：2.7V～5.5V
#     ●プログラマブル割り込み：
#     高精度な温度過熱割り込み 温度上昇 / 温度低下割り込み
#     ●SPI互換インターフェース
#     ●8ピン狭幅SOIC、RoHS準拠パッケージ
#
#     SPI コマンド  Byte
#     0       0（固定）
#     1       0（固定）
#     2       1：温度連続読出  0:
#     [5:3]   7−0:レジスターアドレス
#     6       1:読み出し  0:書き込み
#     7       0（固定）
#
#     コンフィギュレーション・レジスタ（レジスタ・アドレス0x01）
#     [1:0]  これらの2つのビットは、
#          INTピンとCTピンを設定する前に発生する可能性のある低温/過温度フォルトの
数を設定します。
#          これにより、誤ったトリガーを避けることができます
#          温度ノイズ。
#          00 = 1フォルト（デフォルト）。
#          01 = 2フォルト。
#          10 = 3フォルト。
#          11 = 4フォルト。
#     2         このビットは、CTピンの出力極性を選択します。
#          0 =アクティブロー。1 =アクティブハイ。
```

#  3  このビットは、INTピンの出力極性を選択します。
#  0 =アクティブロー。1 =アクティブハイ。
#  4  このビットは、コンパレータモードと割り込みモードを選択します。
#  0 =割り込みモード。1 =コンパレータモード。
#  [6:5]  これら2つのビットは、ADT7310の動作モードを設定します。
#  00 =連続変換（デフォルト）。 1つの変換が終了すると、ADT7310は別の変換を開始します。
#  01 =ワンショット。変換時間は通常240ミリ秒です。
#  10 = 1 SPSモード。変換時間は通常60msです。この動作モードは平均消費電流を低減します。
#  11 =シャットダウン。インターフェイス回路を除くすべての回路がパワーダウンされます。
#  7  このビットは変換時にADCの分解能を設定します。
#  0 = 13ビット分解能。符号ビット+12ビットは0.0625℃の温度分解能を与えます。
#  1 = 16ビット分解能。符号ビット+ 15ビットは0.0078125℃の温度分解能を与えます。
#