對 PLC 的序列埠做壓力測試

- 原本目的：控制PLC(gpu228)的序列埠，透過它不斷傳資料給另一接收端(例如pc的com port)，且確認資料不會遺失
- 測試：資料之間的最快更新時間間隔設為100mS時，發現資料約每兩小時出錯一次
  - 以上錯誤結果在Windows 10作業系統測試下屢試不爽，而且進一步發現出錯時間和資料的更新間隔呈正比(例如當間隔改成50mS時，則變成每1小時出錯一次)
  - 然而，以上錯誤(每2小時)卻不會在linux作業系統上發生，沒發現任何資料遺失或錯誤
    - 即使測試時用到的硬體和程式都一樣：PLC(GPU228)硬體、PLC和PC端的USB轉序列埠傳輸線(USB-PPI)、PLC序列埠控制程式、pc端的壓力測試程式(windows和linux只在開啟序列埠那一行程式寫法不同)
      - 仍舊有差異，因為作業系統不同，所以USB轉SERIAL的驅動程式也不可能一模一樣
  - 在測試過程中，發現序列埠似乎會不斷丟資料出來，無法如預期可以藉由電腦端控制其傳輸速度
    - 可能得透過序列埠的流量控制等較為複雜功能才有辦法解決
- 初步結論：對PLC的serial port操作在只傳出資料流程的掌控度不夠(沒收PC端的資料)

Test Pattern (PC side)

讀取時間間隔 (秒)	最大讀取次數	最大測試時間 (秒)
0.3	1000	300
5	300	1500
1	300	300
3	300	900
0.5	3000	1500
10	300	3000
2	750	1500

共花費 2.5 小時
(當資料沒遺失)

作業系統和序列埠驅動程式的差異

Windows 規格 Windows 10 home 19042.804

版本	Windows 10 家用版
版本	20H2
安裝於	2021/4/1
OS 組建	19042.804
體驗	Windows Feature Experience Pack 120.2212.551.0

USB-SERIAL CH340 (COM3) - 內容

一般 | 連接埠設定 | 驅動程式 | 詳細資料 | 事件

USB-SERIAL CH340 (COM3)

驅動程式提供者:	wch.cn
驅動程式日期:	2014/8/8
驅動程式版本:	3.4.2014.8
數位簽署者:	Microsoft Windows Hardware Compatibility Publisher

Linux Ubuntu 20.04.2 LTS

Version

Kernel	Linux 5.8.0-48-generic (x86_64)
Version	#54~20.04.1-Ubuntu SMP Sat Mar 20 13:40:25 UTC 2021
C Library	GNU C Library / (Ubuntu GLIBC 2.31-0ubuntu9.2) 2.31
Distribution	Ubuntu 20.04.2 LTS

QinHeng Electronics HL-340 USB-Serial adapter
Transcend Information, Inc. RDF8
Linux Foundation 2.0 root hub

Device Information

Product	HL-340 USB-Serial adapter
Vendor	QinHeng Electronics
Max Current	96 mA

Misc

USB Version	1.10
Class	(Unknown)
Vendor ID	0x1a86
Product ID	0x7523
Bus	1

電腦端的壓力測試程式

```python
import serial
import time

ser = serial.Serial('/dev/ttyUSB0')  # open f
#ser = serial.Serial('COM3')  # open first se
print(ser.portstr)       # check which port w

ser.timeout = 0.5 # read timeout?

old = 48
error={}

delay_list = [
               (0.3, 1000),
               (5, 300),
               (1, 300),
               (3, 300),
               (0.5, 3000),
               (10, 300),
               (2, 750)
              ]

for i in delay_list:

    delay = i[0]
    max_cnt = i[1]

    count = 0
    print('RX interval:', delay, 'second(s)')

    while count < max_cnt:
        #ser.write(b'\x31\n')
        time.sleep(delay) # 1, 10, 0.5, 0.1,3
        get = ser.read(1)
        print(get.hex())
        if get == b'':
            print('Standby')
            continue

        count = count + 1

        new = int.from_bytes(get[0:], 'big')
        if new > old:
            if (new - old) != 1:
                print('error', count)
                error[str(delay)] = count
                old = new # add for the next
                break
        else: # new <= old
            if new != 0 or old != 255:
                print('error', count)
                error[str(delay)] = count
                old = new # add for the next
                break
        old = new

print('test patterns:', delay_list)
print('errors:', error)

ser.close()
```

PLC 端的序列埠控制程式

主程式初始化 (只執行一次)

PROGRAM COMMENTS

Network 1 Network Title

Network Comment

```
LD      SM0.1
MOVB    16#09, SMB30
MOVB    100, SMB34
ATCH    INT_1:INT1, 10
MOVB    1, VB200
MOVB    16#31, VB201
MOVB    16#31, VB299
ATCH    INT_2:INT2, 9
ENI
```

← 設定：每隔 **100mS** 會進入中斷 1

← 設定 :VB200 填 1 代表要傳一筆資料
而 VB201 則填入實際要傳的資料

← 設定：傳完一筆資料後會進入中斷 2

主程式無限迴圈 (不斷重複執行)

Network 4

```
LDN     I0.0
=       Q1.0
```

點亮 Q1.0 LED

中斷 1 (INT_1)

Network 1

```
LD      SM0.0
DTCH    10
XMT     VB200, 0
```

暫時關掉 100mS 中斷 1
並觸發一筆資料傳出
(PLC 硬體會等到該筆資料被傳走後才產生中斷 2)

中斷 2 (INT_2)

Network 1

```
LD      SM0.0
INCB    VB299
MOVB    VB299, VB201
ATCH    INT_1:INT1, 10
```

將下次要傳資料的值加 1
並重新啟動 100mS 中斷 1