NuPython (Python API for NuStreams Systems) Programming Guide

Version : 1.6
Date : 2023/11/27
Author : Taken, Shih

Revision History

Date	Version	History
2016/09/06	0.1	First draft version
2016/09/08	0.2	● 修正文档中错别字 ● 新增ClearPortCounter以及ClearStreamCounter ● 移除connect之后去读取board/link/eeprom状态的动作，加快速度。让用户于必要时手动去读取。 ● 修正示例程序的错误(缺漏Unlock())
2016/09/09	0.3	● 新增StopPortCounter / StopStreamCounter ● 新增 ACK 机制，确保命令被确实执行 ● 重新编写2.1节结构，置换图说明
2016/09/23	0.4	● 修正SetRxStreamCounter执行时机 ● 新增GlobalTransmitPacket命令 ● 新增StartRxStream命令
2016/11/30	0.5	● 正名NuPython ● Stream个数缩短改为32条 ● 修改Counter储存方式，读到Counter先原封不动储存。User来要再转换，节省时间 ● 简化TxStreamMAP设置 ● 新增可产生ARP protocol封包
2016/12/20	0.6	● 修正所有函数命名，使符合python规范 ● 修正read_counter_stream_start参数错误 ● 新增media type setting功能 ● 所有函数增加引数，范例说明
2016/12/22	0.7	● 新增函数命名规则章节
2017/06/08	0.8	● 新增capture命令 ● 新增输出封包详细信息与内容 ● 补充引用函数说明 ● 补充capture并呈现分析结果实例 ● 补充UI呈现实例(内件函式)
2017/07/31	0.9	● 修正结构图 ● 新增章节11 - 板卡Auto ARP Reply / Ping / DHCP Client的设置以及功能 ● 补充ping(含arp)以及dhcp脚本
2017/09/29	1.0	● 新增NuPOE(T451)-章节12 ● 新增NuPOE(T451)demo脚本-章节13.5
2021/09/03	1.1	● 修正命令名称错误:get_counter_port() ● 修正遗漏命令: read_counter_port_stop()
2021/12/07	1.2	● 新增config_stream_adderror命令，使RM板卡可以送出特定错误包
2022/04/18	1.3	● 修改config_stream_pktlen(length)最大長度為16K ● 新增端口Tx flowctrl設定: config_port_flowctrl_tx(enable) ● 新增端口Rx flowctrl設定: config_port_flowctrl_rx(enable)
2023/11/06	1.4	● 修正config_tx_txtime(val)函數錯誤

		● 修正transmit_pkts_sync()函數沒作用 ● 修正config_stream_enable_randomlen(val)函數錯誤 ● 新增多組counter report index ● 新增get_modelname() ● 新增XTAG相關設定config_stream_enable_xtag(val)
2023/11/20	1.5	● 修正read_info_board多個槽位錯誤問題 ● 修正read_license_board多個槽位錯誤問題 ● 新增CallBack Function應用（Ch.13）
2023/11/27	1.6	● 新增transmit_pkts_sync_stop()函數與說明 ● 新增transmit_pkts_stop()說明

目录

1. Quick Start

NuPython 是运行在 Python 环境底下，操作 NuStreams 测试系统的开源脚本，其中 Python 为免费软件，自行下载安装 Python 环境之后，将[NuPython.py]放置 python 环境底下，即可开始使用。本 API 不限定 Python 运行版本(Python2/3 适用)，不限定作业系统(Windows/Linux 适用)。

使用时运用命令:

import NuPython

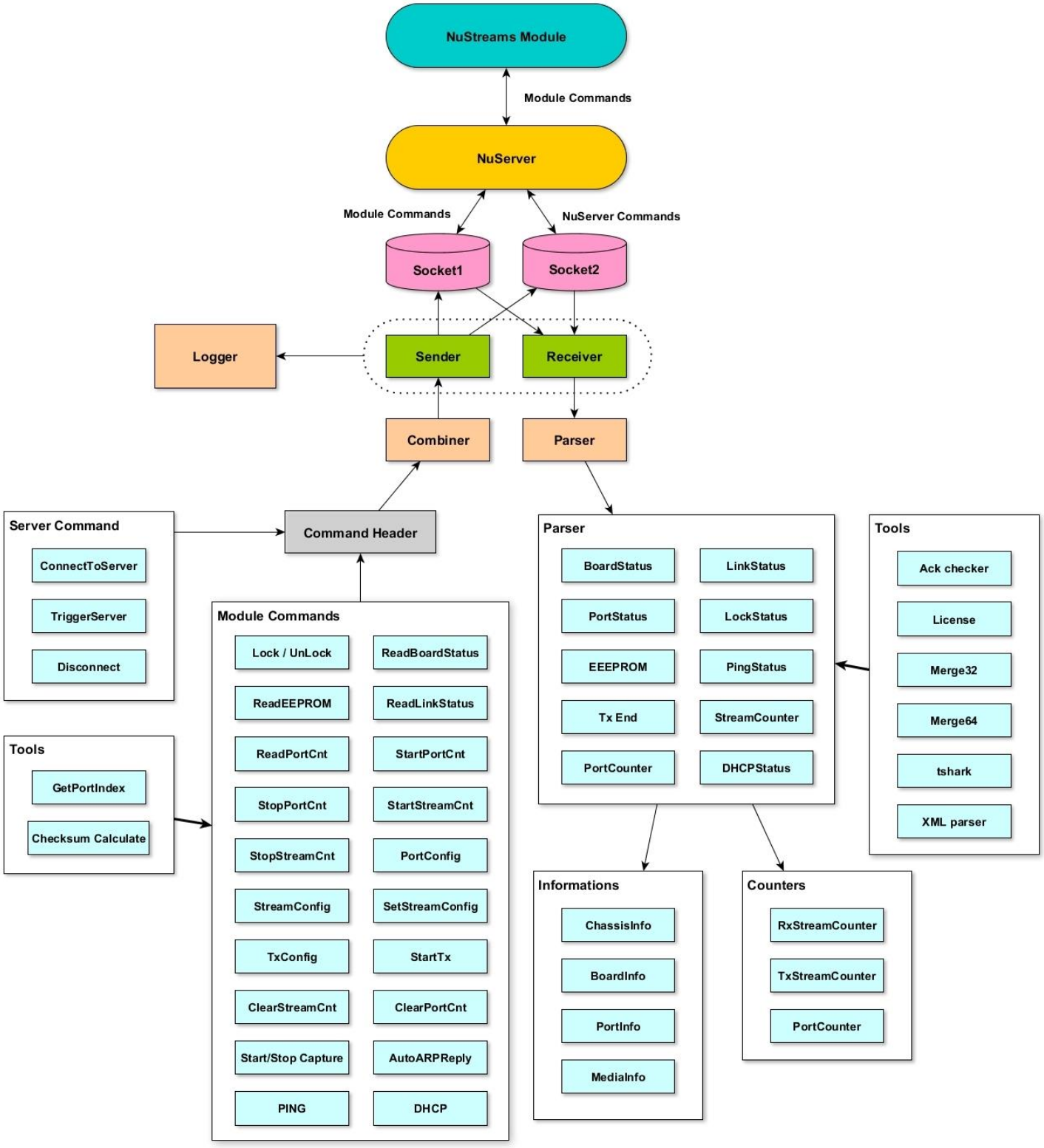
本文章接下来介绍 NuPython 的架构，并提供编程建议。

2. Programming

本章节主要介绍编程步骤。首先会讨论 NuPython 整个结构说明，再来定义 NuPython 的命名规则。最重要是用一个简单实例，从无到有利用 NuPython 一步步建立一个小程序。最后列举一下 NuPython 引用那些的外部功能。

2.1 NuPython 结构与规范

NuPython 提供的命令以及命令之间的关系图，如图下所示。NuPython 命令，使用前必须先和 Server 连线用 socket 方式连接收送。这边开启两个 socket 和 server 连线，一为 Port = 4000(socket1)，一为 Port = 4001(socket2)。其中 Socket2 负责和 Server 沟通保持连线，程序中每个命令之后必须搭配一个 trigger Server 命令，Server 才会持续处理下一步动作。Socket1 负责其他所有命令。每个命令还包含其他子命令，将在下一章说明。



- **NuStreams Module**
  NuStreams 控制卡
- **NuServer**
  统整命令，并和 NuStreams Module 沟通的 AP。NuStreams 600i/2000i 系列，NuServer 为独立 AP，需要额外启动。NuStreams 700 系列，NuServer 内建于控制卡，用户不用个别启动，只需以 IP 连线即可。
- **Logger**

无论收送命令，或是其他内部动作，皆自动写出文档，供 debug 使用。因此最终需要使用关档命令结束。

- **Combiner**
  将 Header 和 Commmand 合并成为一个命令。透过 Socket 送出。因为每个 NuStreams 命令，皆有标头档，NuPython 特别将标头档拆开，可以独立设置。最后再透过 combiner 组合起来即为完整命令。
- **Parser**
  接收到来自 Socket 命令后，根据不同 ID，选择不同 Parser 动作，再将处理结果放入 Counters，供后续读取。比方收到 board 信息 id，NuPython 会丢给 board info parser 分析，之后再将分析结果丢到 board info 列表保存。
- **Command Header**
  所有传送命令，统一由 Command Header 加上标头，统一管理 ID，sequence number。
- **Server Command**
  提供针对 Server 相关的命令，比方 connect/disconnect/trigger 等命令，这些命令皆透过 Socket2 传送给 Server。
- **Module Command**
  提供除了 Server 命令之外，其余板卡相关的命令，这些命令透过 Socket1 传送给 Server。Server 再转换后送给 Module
- **Tools**
  提供一些常用工具，比方取得 Port 在 List 中的 Index，比方说 Checksum 的计算。
- **Information**
  存放系统，板卡等相关信息，如版本号，MAC，Serial Number，License 等等。
- **Counter**
  存放板卡及板卡特定 Stream 结果数据。

其余详细内容参照之后章节。

2.2 函数命名规则

NuPython 函数皆符合 Python 命名规范，NuPython 没有全局变量，所有变量皆透由函数操作。NuPython 的函数命名有四种规则如下:

1. **动词_名词**
   此类型函数命名较为直觉，大部份用于主要设置。比方”传送封包”等命令:
   - set_media
   - set_stream
   - transmit_pkts
2. **名词_动词**
   此类型的函数，是在这个名词有很多行为动作时使用，名词用于前方，可以方便排序。比方，连接"Server"，断开”Server”，触发”Server”，三者皆有 Server 这个名词，因此把名词放于前方，函数命名如:
   - server_connect

- server_disconnect
- server_trigger

3. **动词_形容词_名词**

   此类型函数命名如同「动词_名词」，属直觉上的命名。比方，”配置_传送的_包数”，或是”配置_stream 的_包长”等，如:

   - config_stream_pktlen
   - config_tx_txpkts

4. **动词_名词_形容词**

   此类型函数类似「名词_动词」的倒置语气方式，名词和形容词对调，目的是方便排序。比方:读取 port 的”counter”，读取 stream 的”counter”，都有 counter，因此 counter 提前。又或者是读 取 board 的”信息”，读取 link 的”信息”，都有信息，因此提前:

   - read_counter_port
   - read_counter_stream
   - read_info_board
   - read_info_license

2.3 编程步骤

以下提供简单的步骤，编写一个可执行的 Python 脚本

1. **载入 NuPython.py，适时调用 nuconst.py(常数变量)**

   在 python 脚本中，优先载入，就可以使用，格式如下:

   import NuPython

   其中，NuStreamsModuleSetting()是主要类别，我们可以先用别名宣告，方便后面程序使用

   nscmd = NuPython.NuStreamsModuleSetting()

2. **连接到 NuServer**

   NuStreams 600i/2000i 机箱，必须先开启 NuServer，或是 NuServer 于远端(遠端 600/2000 機箱或 700/900 机箱直連)。NuServer 准备好之后，透过下面命令连线到 NuServer

   nscmd.server_connect("127.0.0.1")

   连线之后，需要等待 2-3 秒，因为自动去把重要信息搜集，比方 BoardStatus/LinkStatus/EEPROM Report 等等

3. **锁定端口**

   将需要的端口给保留，避免其它人使用。首先需要将端口给标记，标记哪些是需要的。标记完之后，再给端口锁定。

   nscmd.port_mark(chassisID1, boardID1, portID1)
   nscmd.port_mark(chassisID2, boardID2, portID2)
   ……
   nscmd.port_lock()

4. **获取端口 index**

   因为脚本里头的程序，都是用 port index 为参数，比起(chassis, board, port)组合，节省了许多输入上问题，这边也提供一个工具可以查找:

port1_idx = nscmd.get_portidx(chassisID1, boardID1, portID1)

5. 配置传输埠

nscmd.config_stream_pktlen(pkt_len)
nscmd.config_stream_enable_randomlen(random_len)
 # protocol
nscmd.config_stream_enable_vlan(0)
 # 配置 MAC
nscmd.config_stream_smac("00:22:A2:00:03:01")
nscmd.config_stream_dmac("00:22:A2:00:03:02")
 # 配置 IP
nscmd.config_stream_sip("192.168.3.1")
nscmd.config_stream_dip("192.168.3.2")
 # 配置 UDP Port
nscmd.config_stream_sport(100)
nscmd.config_stream_dport(1000)
nscmd.config_stream_utilization(80)
 # protocol=1 means layer3, 2 means udp
nscmd.config_stream_protocol(2)
 # set total streams number
nscmd.config_stream_streamnum(1)
 # 设置传输埠
nscmd.set_stream(port1_idx, 0)

6. 开始传送

 # 配置传送的时间或是传送的个数，2 选 1
nscmd.config_tx_txtime(tx_time)
nscmd.config_tx_txpkts(packets)
 # 开始打包
nscmd.transmit_pkts(port1_idx)

7. 读取结果

#读取端口 Counter
nscmd.get_counter_port(port1_idx, nscmd.IDX_PORTCOUNTER_RX_GOODPKT)
#读取端口 Stream Counter
nscmd.get_counter_stream_rx_pkts (port1_idx, 0)

8. 端口解锁

nscmd.port_unlock()

9. 和 NuServer 断开连线

nscmd.server_disconnect()

2.4 引用到的函数

2.4.1 Python 内建函数

- socket - 与 Server 传送和接收信息
- struct - 组合与拆解网络包
- sys - 判断 Python 版本
- time - 动态生成当前时间，提供给 log 使用
- array - 板卡 map 组合
- os - 可以载入外部程序
- xml.sax - XML constructor / parser
- Tkinter - UI 元件
- tkMessageBox - 弹出对话框

2.4.2 外部函数及程序

必须安装 wireshark，因为程序会使用到里头 tshark 分析工具。其中 processpcap.bat 记载 wireshark 安装路径，可以修改以符合所需。目前分析功能只限定 Windows 用户，Linux 仍在研究。

3. 一般命令

本章节为列举所有 NuPython 和版卡无关的功能函数，包括 Server 连线断线，保留释放端口，以及取得端口在端口列表中的 index。之后和端口的命令，都是以 index 为主，因此在操作时，都得先取得端口 index。

3.1 server_connect(ip)

Function	和 NuServer 以 IP 连线
Input	IP。String
Output	None
Note	
Example	*ip = 192.168.1.1* *self.server_connect(ip)*

3.2 server_disconnect()

Function	断开和 NuServer 连线
Input	None
Output	None
Note	
Example	*self.server_disconnect()*

3.3 port_mark(chassis, board, port) / port_unmark(chassis, board, port)

Function	标记/解除标记需要保留 port
Input	Chassis ID, Board ID, Port ID
Output	None
Note	
Example	*chassis = 0* *board = 2* *port = 1* *self.port_mark(chassis, board, port)* *self.port_unmark(chassis, board, port)*

3.4 port_lock() / port_unlock()

Function	锁定/解锁刚刚标记(port_mark)的 port
Input	None
Output	None
Note	配合 port_mark/port_unmark 运作
Example	*self.port_lock()* *self.port_unlock()*

3.5 get_portidx(chassis, board, port)

Function	取得 port 相对应的 index
Input	ChassisID, BoardID, PortID
Output	Port index
Note	
Example	*chassis = 0* *board = 2* *port = 1* *pidx = self.get_portidx(chassis, board, port)*

4 读取信息命令

本章节说明所有获取信息的命令。包含板卡信息，端口连结状态，端口收发包计数等等。

4.1 read_info_board(pidx)

Function	取得板卡的版本号信息
Input	Port index
Output	None
Note	利用端口 1 的 pidx 呼叫，可再搭配 get_version_hw/get_version_fw/get_version_prom/get_modelname 使用
Example	*pidx = self.get_portidx(0,2,1)* *self.read_info_board(pidx)* *print (" hw ver. = ", nscmd.get_version_hw(bid))* *print (" fw ver. = ", nscmd.get_version_fw(bid))* *print (" prom ver. = ", nscmd.get_version_prom(bid))* *print (" cardtype = ", nscmd.get_modelname(bid))*

4.2 read_license_board(pidx)

Function	取得板卡的 eeprom 中的 license 信息
Input	Port index
Output	None
Note	透過端口 1 的 pidx 呼叫，可再搭配 get_license_mode 以及 get_license_date 使用
Example	*pidx = self.get_portidx(0,2,1)* *self.read_license_board(pidx)* *print (" MAC addr = ", nscmd.get_macaddr(bid))* *print (" serial number = ", nscmd.get_serialnum(bid))* *print (" manual date = ", nscmd.get_manudate(bid))* *print (" license mode = ", nscmd.get_license_mode(bid))* *print (" license date = ", nscmd.get_license_date(bid))*

4.3 read_info_link(pidx)

Function	取得 port 的 Link 状态
Input	Port index
Output	None

Note	可再搭配 get_media_speed/get_media_duplex/get_media_autonego 使用
Example	*pidx = self.get_portidx(0,2,1)* *self.read_info_link(pidx)* *print ("Media speed = " + nscmd.get_media_speed(pidx))* *print ("Media duplex = " + nscmd.get_media_duplex(pidx))* *print ("Media negotiation = " + nscmd.get_media_autonego(pidx))*

4.4 read_counter_port_once(pidx)

Function	取得 port 的 Tx/Rx 数值
Input	Port index
Output	None
Note	
Example	*pidx = self.get_portidx(0,2,1)* *self.read_counter_port_once(pidx)*

4.5 read_counter_port_start(pidx, interval)

Function	间隔时间固定自动取得 port 的 Tx/Rx 数值
Input	Port index, interval
Output	None
Note	Interval 为间隔时间，单位为秒。标记每间隔时间返回依次数值
Example	*pidx = self.get_portidx(0,2,1)* *interval = 2* *self.read_counter_port_start(pidx, interval)*

4.6 read_counter_port_stop(pidx)

Function	停止间隔时间回報 port 的 Tx/Rx 数值
Input	Port index
Output	None
Note	
Example	*pidx = self.get_portidx(0,2,1)* *interval = 2* *self.read_counter_port_start(pidx, interval)* *self.read_counter_port_stop(pidx)*

4.7 read_counter_stream_once(pidx)

Function	间隔时间固定自动取得 stream 的 Rx 数值

Input	Port index, Stream index
Output	None
Note	
Example	*pidx = self.get_portidx(0,2,1)* *self.read_counter_stream_once(pidx)*

4.8 read_counter_stream_start(pidx, interval)

Function	间隔时间固定自动取得 port 的某条 stream 的 Tx/Rx 数值
Input	Port index，interval
Output	None
Note	Interval 为间隔时间，单位为秒。标记每间格时间返回依次数值
Example	*pidx = self.get_portidx(0,2,1)* *interval = 2* *self.read_counter_stream_start(pidx, interval)*

5 切换 Media 命令

本章节说明个别端口 media 的设置，包含速率，全/半双工，自动协商，媒介设置等等。设置后，从外观上可以看见(部分)板卡会先熄灯(link-down)，然后重新做协商配对，等到完全稳定，会再次点亮 led 灯。过程约 3~5 秒不等，也有可能一直熄灯点灯循环，此时表示无法连结稳定，可以改用非自动设置方式。

5.1 config_media_speed(speed)

Function	设置 media speed
Input	speed
Output	None
Note	speed - ● MEDIA_SPEED_10G ● MEDIA_SPEED_1G ● MEDIA_SPEED_100M ● MEDIA_SPEED_10M
Example	*self.config_media_speed(self.MEDIA_SPEED_1G)*

5.2 config_media_duplex(duplex)

Function	设置 media duplex，全双工或半双工
Input	duplex
Output	None
Note	duplex - ● MEDIA_DUPLEX_FULL ● MEDIA_DUPLEX_HALF
Example	*self.config_media_duplex(self.MEDIA_DUPLEX_FULL)*

5.3 config_media_autonego(negotiation)

Function	设置 media negotiation，自动查找或强制设定
Input	negotiation
Output	None
Note	negotiation - ● MEDIA_NEGO_AUTO ● MEDIA_NEGO_FORCE
Example	*self.config_media_*autonego*(self.MEDIA_NEGO_FORCE)*

5.4 config_media_signal(signal)

Function	设置 media signal，光口或电口
Input	signal
Output	None
Note	Signal - ● MEDIA_SIGNAL_FIBER ● MEDIA_SIGNAL_COPPER
Example	*self.config_media_signal(self.MEDIA_SIGNAL_FIBER)*

5.5 set_media(pidx)

Function	根据 5.1-5.4 配置，设置至特定端口
Input	Port index
Output	None
Note	
Example	*pidx = self.get_portidx(0,2,1)* *self.set_media(pidx)*

6 数据清除，停止命令

清除端口收发包信息量。当下了此命令当下，端口的收发包数据当下会被清为 0，若是持续不断的发包，清为 0 仅瞬间。用户通常不用另外下此命令。会在 transmit_pkts 里头自动运行。

6.1 clear_counter_port(pidx)

Function	清除端口的 Counter 数据。
Input	Port Index
Output	None
Note	
Example	*pidx = self.get_portidx(0,2,1)* *self.clear_counter_port(pidx)*

6.2 clear_counter_stream(pidx)

Function	清除端口所有 Stream 的 Counter 数据。
Input	Port Index
Output	None
Note	
Example	*pidx = self.get_portidx(0,2,1)* *self.clear_counter_stream(pidx)*

6.3 read_counter_stop()

Function	停止所有端口，所有 Stream 的 Counter 数据回报。
Input	None
Output	None
Note	自动查找有被 lock 的端口，并下达停止回报命令。包含 ReadPortCounterStop 以及 ReadStreamCounterStop 命令。在 unlock 时自动呼叫。
Example	*self.read_counter_stop()*

7 配置端口相关命令

大部分都为非端口相关的设置，为单一变数，保存在 python 文档中，设置最后呼叫[set_stream]，即完成端口传送相关的配置。

7.1 config_stream_streamnum(number)

Function	设置端口 stream 数量。设置后，会自动分配 stream index 给相关命令使用，用户只需关心 stream 数量就好。
Input	Value
Output	None
Note	数值范围为 1～32
Example	*streamnum = 16* *self.config_stream_streamnum(streamnum)*

7.2 config_stream_utilization(loading)

Function	设置端口 stream 的 rate。单位为 100%，设置之后，会自动根据 speed 等参数，计算出对应的 frame gap。
Input	Value
Output	None
Note	数值范围为 1～100
Example	*loading = 50* *self.config_stream_utilization(loading)*

7.3 config_stream_enable_randomlen(enable)

Function	开启/关闭端口 stream 的 Random 包长度。1 为开启，0 为关闭。
Input	Value
Output	None
Note	数值范围为 0 和 1
Example	*is_enable = 0* *self.config_stream_enable_randomlen(is_enable)*

7.4 config_stream_pktlen(length)

Function	设置端口 stream 的包长度

Input	Value
Output	None
Note	数值范围为 48～16000
Example	*pkt_len = 64* *self.config_stream_pktlen(pkt_len)*

7.5 config_stream_streamid(sid)

Function	设置端口 stream 的 ID。用户不用填写，此为脚本根据设置的 stream index 自动填写（streamid = stream index+1）
Input	Value
Output	None
Note	数值范围为 1～32
Example	*stream_id = 1* *self.config_stream_streamid(stream_id)*

7.6 config_stream_enable_vlan(enable)

Function	开启/关闭端口 stream 的 VLAN tag。1 为开启，0 为关闭。
Input	Value
Output	None
Note	数值范围为 0 和 1。VLAN Tag 订为 0x8100
Example	*is_enable = 0* *self.config_stream_enable_vlan(is_enable)*

7.7 config_stream_vlan_id(vid)

Function	若 Enable VLAN 有开启的话，此命令设置端口 stream 的 VLAN ID。
Input	Value
Output	None
Note	数值范围为 0～4095
Example	*vid = 100* *self.config_stream_vlan_id(vid)*

7.8 config_stream_vlan_pri(priority)

Function	若 EnableVLAN 有开启的话，此命令设置端口 stream 的 VLAN 优先权。
Input	Value
Output	None
Note	数值范围为 0～7

Example	*priority = 3* *self.config_stream_vlan_pri(priority)*

7.9 config_stream_protocol(protocol_id)

Function	设置端口 stream 的封包格式。
Input	Value
Output	None
Note	protocol_id - 0 : layer2 – 1 : layer3 ipv4 – 2 : layer3 udp – 3 : layer3 ipv6(未实现) – 4 : ARP
Example	*protocol_id = 0* *self.config_stream_protocol(protocol_id)*

7.10 config_stream_smac(mac)

Function	设置端口 stream 的 Source MAC。
Input	Value
Output	None
Note	格式固定为”XX:XX:XX:XX:XX:XX”，以冒号区隔
Example	*mac_addr = “00:22:A2:00:03:01”* *self.config_stream_smac(mac_addr)*

7.11 config_stream_dmac(mac)

Function	设置端口 stream 的 Destination MAC。
Input	Value
Output	None
Note	格式固定为”XX:XX:XX:XX:XX:XX”，以冒号区隔
Example	*mac_addr = “00:22:A2:00:03:01”* *self.config_stream_dmac(mac_addr)*

7.12 config_stream_arp_smac(mac)

Function	设置端口 stream 的 ARP Source MAC。
Input	Value
Output	None
Note	格式固定为”XX:XX:XX:XX:XX:XX”，以冒号区隔

Example	*mac_addr = “00:22:A2:00:03:01”* *self.config_stream_arp_smac(mac_addr)*

7.13 config_stream_arp_dmac(mac)

Function	设置端口 stream 的 ARP Destination MAC。
Input	Value
Output	None
Note	格式固定为”XX:XX:XX:XX:XX:XX”，以冒号区隔
Example	*mac_addr = “00:22:A2:00:03:01”* *self.config_stream_arp_dmac(mac_addr)*

7.14 config_stream_arp_sip(ip)

Function	设置端口 stream 的 ARP Source IP。
Input	Value
Output	None
Note	格式固定为”XXX.XXX.XXX.XXX”，以”.”区隔
Example	*ip = “192.168.1.1”* *self.config_stream_arp_sip(ip)*

7.15 config_stream_arp_dip(ip)

Function	设置端口 stream 的 ARP Destination IP。
Input	Value
Output	None
Note	格式固定为”XXX.XXX.XXX.XXX”，以”.”区隔
Example	*ip = “192.168.1.1”* *self.config_stream_arp_dip(ip)*

7.16 config_stream_sip(ip)

Function	设置端口 stream 的 Source IP。config_stream_protocol 设为 1/2 时适用
Input	Value
Output	None
Note	格式固定为”XXX.XXX.XXX.XXX”，以”.”区隔
Example	*ip = “192.168.1.1”* *self.config_stream_sip(ip)*

7.17 config_stream_dip(ip)

Function	设置端口 stream 的 Dest. IP。config_stream_protocol 设为 1/2 时适用
Input	Value
Output	None
Note	格式固定为”XXX.XXX.XXX.XXX”，以”.”区隔
Example	*ip = “192.168.1.1”* *self.config_stream_dip(ip)*

7.18 config_stream_sport(port_num)

Function	设置端口 stream 的 Source Port。config_stream_protocol 设为 2 时适用
Input	Value
Output	None
Note	
Example	*udp_port = 1000* *self.config_stream_sport(udp_port)*

7.19 config_stream_dport(port_num)

Function	设置端口 stream 的 Dest. Port。config_stream_protocol 设为 2 时适用
Input	Value
Output	None
Note	
Example	*udp_port = 1000* *self.config_stream_dport(udp_port)*

7.20 config_stream_adderror (errorcode)

Function	添加端口 stream 的错误形态
Input	Value
Output	None
Note	Error code: ns_const.ERROR_NO = 0(initial) ns_const.ERROR_CRC = 1 ns_const.ERROR_DRIBBLE = 2(10G 不支持) ns_const.ERROR_ALIGN = 3(10G 不支持)
Example	*err_code = ns_const.ERROR_DRIBBLE* *self.config_stream_adderror(err_code)*

7.21 config_stream_enable_xtag (enable)

Function	开启/关闭端口 stream 的 X-tag。1 为开启，0 为关闭。
Input	Value
Output	None
Note	数值范围为 0 和 1。
Example	*is_enable = 0* *self.config_stream_enable_xtag(is_enable)*

7.22 set_stream(pidx, sidx)

Function	真正将以上变量，设置到端口指定 stream 中。
Input	Port index，Stream index
Output	None
Note	根据 stream index 自动填写 stream id，用意是方便 stream report 设置与管理。 并会算出 frame gap，包数/每秒。
Example	*pidx = self.get_portidx(0,2,1)* *sidx = 3* *self.set_stream(pidx, sidx)*

8 传送相关命令

根据脚本设置，传送可分为两种方式，一种为根据包数传送，一种为根据时间传送。用户设定任一种，就会以此种方式传送。无论哪一种其实都是设置包数，只是以时间为主的，会利用先前计算出的包数/每秒，乘上秒数得到。

8.1 config_tx_txtime(seconds)

Function	设置传送端口的总秒数
Input	Value
Output	None
Note	根据 stream 设置的[包数/每秒]乘上秒数而得到
Example	*tx_time = 10* *self.config_tx_txtime(tx_time)*

8.2 config_tx_txpkts(packets)

Function	设置传送端口的总包数
Input	Value
Output	None
Note	会忽略先前计算得到的[包数/每秒]
Example	*tx_pkts = 100000* *self.config_tx_txpkts(tx_pkts)*

8.3 config_tx_isimmediate(enable)

Function	设置传送端口是否立即传送
Input	enable 0 - wait for a global transmit command 1 - transmit immediatly
Output	None
Note	Default is 1
Example	*is_transmit = 1* *self.config_tx_isimmediate(is_transmit)*

8.4 config_port_flowctrl_tx(enable)

Function	设置传送端口 Tx Flow Control 是否開啟

Input	enable 0 – disable(default) 1 – enable
Output	None
Note	Default is 0。因為屬於端口傳送相關，故放在此章節
Example	*is_flowctrl = 1* *self.config_port_flowxtrl_tx(is_flowctrl)*

8.5 config_port_flowctrl_rx(enable)

Function	设置传送端口 Rx Flow Control 是否開啟
Input	Enable 0 – disable(default) 1 – enable
Output	None
Note	Default is 0。因為屬於端口傳送相關，故放在此章節
Example	*is_flowctrl = 1* *self.config_port_flowxtrl_rx(is_flowctrl)*

8.6 set_config_rxstream(pidx)

Function	设置接收端口的 Stream 配置
Input	Port index
Output	None
Note	必须先于 transmit_pkts 设置， Rx Stream 才会正常运作
Example	*pidx = self.get_portidx(0,2,1)* *self.set_config_rxstream(pidx)*

8.7 transmit_pkts(pidx)

Function	端口开始传送包
Input	Port index
Output	None
Note	根据 config_tx_isimmediate 来决定是否立即传送
Example	*pidx = self.get_portidx(0,2,1)* *self.transmit_pkts(pidx)*

8.8 transmit_pkts_stop(pidx)

Function	端口停止传送包

Input	Port index
Output	None
Note	
Example	*pidx = self.get_portidx(0,2,1)* *self.transmit_pkts_stop(pidx)*

8.9 transmit_pkts_sync()

Function	端口开始同步传送包
Input	None
Output	None
Note	不用设置任何参数，所需参数会在 transmit_pkts 自动取得
Example	*self.transmit_pkts_sync()*

8.10 transmit_pkts_sync_stop()

Function	端口停止同步传送包
Input	None
Output	None
Note	不用设置任何参数，所需参数会在 transmit_pkts 自动取得
Example	*self.transmit_pkts_sync_stop()*

9 结果数值命令

获取板卡或端口相关数值的命令。通常都需要有先前的获取命令，然后板卡回覆后，经由 Parser 分析后储存，再透由此结果数值命令呈现。

9.1 Module 相关

获取 module 相关信息，其中 8.1.1~8.1.3 需要先下 read_info_board 命令之后再读取，8.1.4~8.1.8 需要先下 read_info_eeprom 命令之后再获取。

9.1.1 get_version_hw(slotid)

Function	获取模块的硬件版本号

Input	Slot ID
Output	Hardware version
Note	如果输入不存在 slot，则显示空值。
Example	*slotid = 5*
	version = self.get_version_hw(slotid)

9.1.2 get_version_fw(slotid)

Function	获取模块的韧件版本号
Input	Slot ID
Output	Firmware version
Note	如果输入不存在 slot，则显示空值
Example	*slotid = 5*
	version = self.get_version_fw(slotid)

9.1.3 get_version_prom(slotid)

Function	获取模块的 PROM 版本号
Input	Slot ID
Output	PROM version
Note	如果输入不存在 slot，则显示空值
Example	*slotid = 5*
	version = self.get_version_prom(slotid)

9.1.4 get_serialnum(slotid)

Function	获取模块的序号
Input	Slot ID
Output	Serial Number
Note	如果输入不存在 slot，则显示空值
Example	*slotid = 5*
	serialnum = self.get_serialnum(slotid)

9.1.5 get_macaddr(slotid)

Function	获取模块的 MAC id
Input	Slot ID
Output	MAC address
Note	如果输入不存在 slot，则显示空值
Example	*slotid = 5*

	mac_addr = self.get_macaddr(slotid)

9.1.6 get_manudate(slotid)

Function	获取模块的出厂日期
Input	Slot ID
Output	Manufacture Date
Note	如果输入不存在 slot，则显示空值
Example	*slotid = 5*
	manu_date = self.get_manudate(slotid)

9.1.7 get_license_mode(slotid)

Function	获取模块的授权模式
Input	Slot ID
Output	License Mode。Normal/ Evaluation
Note	如果输入不存在 slot，则显示空值
Example	*slotid = 5*
	lic_mode = self.get_license_mode(slotid)

9.1.8 get_license_date(slotid)

Function	获取模块的授权日期
Input	Slot ID
Output	License Date
Note	如果输入不存在 slot，则显示空值
Example	*slotid = 5*
	lic_mode = self.get_license_date(slotid)

9.1.9 get_modelname(slotid)

Function	获取模块的中文名稱
Input	Slot ID
Output	Model Name
Note	如果输入不存在 slot，则显示空值
Example	*slotid = 5*
	name_mode = nscmd.get_modelname(slotid)

9.2 Link 相关

需要先下 read_info_link 命令获取。

9.2.1 get_media_speed(pidx)

Function	获取端口的 Media 速率
Input	Port index
Output	MEDIA_SPEED_5G = 5 MEDIA_SPEED_2P5G = 4 MEDIA_SPEED_10G = 3 MEDIA_SPEED_1G = 2 MEDIA_SPEED_100M = 1 MEDIA_SPEED_10M = 0 MEDIA_SPEED_LINKDOWN = 0xff
Note	
Example	*pidx = self.get_portidx(0,2,1)* *media_speed = self.get_media_speed(pidx)*

9.2.2 get_media_duplex(pidx)

Function	获取端口的 Media 双工模式
Input	Port index
Output	Full / Half
Note	全双工或半双工
Example	*pidx = self.get_portidx(0,2,1)* *media_duplex = self.get_media_duplex(pidx)*

9.2.3 get_media_autonego(pidx)

Function	获取端口的 Media 协商模式
Input	Port index
Output	Auto / Force
Note	自动或是手动协商模式
Example	*pidx = self.get_portidx(0,2,1)* *media_autonego = self.get_media_autonego(pidx)*

9.2.4 get_media_signal(pidx)

Function	获取端口的 Media 信号模式
Input	Port index
Output	Copper / Fiber
Note	光口或电口
Example	*pidx = self.get_portidx(0,2,1)* *media_signal = self.get_media_signal(pidx)*

9.3 Port Counter 相关

9.3.1 get_counter_port(pidx, countidx)

Function	获取端口的封包数，根据 counter index 获取相对应数据
Input	Port index, Counter index
Output	Packets number
Note	countidx ● IDX_PORTCOUNTER_RX_BORADCAST ● IDX_PORTCOUNTER_RX_MULTICAST ● IDX_PORTCOUNTER_RX_UNICAST ● IDX_PORTCOUNTER_RX_PAUSE ● IDX_PORTCOUNTER_RX_VLAN ● IDX_PORTCOUNTER_RX_IPV4 ● IDX_PORTCOUNTER_RX_ERR_DRIBBLE ● IDX_PORTCOUNTER_RX_ERR_ALIGN ● IDX_PORTCOUNTER_RX_ERR_CRC ● IDX_PORTCOUNTER_RX_UNDERSIZE ● IDX_PORTCOUNTER_RX_OVERSIZE ● IDX_PORTCOUNTER_RX_GOODPKT ● IDX_PORTCOUNTER_RX_ERR_DI ● IDX_PORTCOUNTER_RX_ERR_IPCHKSUM ● IDX_PORTCOUNTER_RX_64 ● IDX_PORTCOUNTER_RX_65_127 ● IDX_PORTCOUNTER_RX_128_255 ● IDX_PORTCOUNTER_RX_256_511 ● IDX_PORTCOUNTER_RX_512_1023 ● IDX_PORTCOUNTER_RX_1024_1522 ● IDX_PORTCOUNTER_RX_CAPTURE

	• IDX_PORTCOUNTER_RX_HOSTQUEFULL • IDX_PORTCOUNTER_RX_ICMP • IDX_PORTCOUNTER_RX_ARP • IDX_PORTCOUNTER_RX_ERR_FRAGMENT • IDX_PORTCOUNTER_RX_ERR_TCPCHKSUM • IDX_PORTCOUNTER_RX_ERR_UDPCHKSUM • IDX_PORTCOUNTER_RX_IPV4FRAGMENT • IDX_PORTCOUNTER_RX_IPV4EXTENSION • IDX_PORTCOUNTER_RX_XTAG • IDX_PORTCOUNTER_RX_GAPOVER12 • IDX_PORTCOUNTER_RX_BYTE • IDX_PORTCOUNTER_RX_RATE_BYTE • IDX_PORTCOUNTER_RX_RATE_PKT • IDX_PORTCOUNTER_TX_PKT • IDX_PORTCOUNTER_TX_COL • IDX_PORTCOUNTER_TX_COL_MULTI • IDX_PORTCOUNTER_TX_COL_EXC • IDX_PORTCOUNTER_TX_PAUSEPACKET • IDX_PORTCOUNTER_TX_BYTE • IDX_PORTCOUNTER_TX_COL_TOTAL • IDX_PORTCOUNTER_TX_RATE_BYTE • IDX_PORTCOUNTER_TX_RATE_PKT
Example	*pidx = self.get_portidx(0,2,1)* *countidx = self.IDX_PORTCOUNTER_RX_GOODPKT* *counter = self.get_counter_port(pidx, countidx)*

9.4 Stream Counter 相关

9.4.1 get_counter_stream_rx_pkts(pidx, sidx)

Function	根据 Port index 以及 stream index 获取端口特定 stream 的接收包数
Input	Port index, Stream index
Output	Packets number
Note	
Example	*pidx = self.get_portidx(0,2,1)* *sidx = 0* *counter_pkts = self.get_counter_stream_rx_pkts(pidx, sidx)*

9.4.2 get_counter_stream_rx_bytes(pidx, sidx)

Function	根据 Port index 以及 stream index 获取端口特定 stream 的接收包字节数
Input	Port index, Stream index
Output	Bytes number
Note	
Example	*pidx = self.get_portidx(0,2,1)* *sidx = 0* *counter_bytes = self.get_counter_stream_rx_bytes(pidx, sidx)*

9.4.3 get_counter_stream_rx_latency(pidx, sidx)

Function	根据 Port index 以及 stream index 获取端口特定 stream 的时延
Input	Port index, Stream index
Output	时延值
Note	此值必须乘上 0.04，单位为 microsecond。
Example	*pidx = self.get_portidx(0,2,1)* *sidx = 0* *latency = self.get_counter_stream_rx_latency(pidx, sidx)*

9.4.4 get_counter_stream_tx_pkts(pidx, sidx)

Function	根据 Port index 以及 stream index 获取端口特定 stream 的传送包数
Input	Port index, Stream index
Output	Packets number
Note	
Example	*pidx = self.get_portidx(0,2,1)* *sidx = 0* *counter_pkts = self.get_counter_stream_tx_pkts(pidx, sidx)*

9.4.5 get_counter_stream_tx_bytes(pidx, sidx)

Function	根据 Port index 以及 stream index 获取端口特定 stream 的传送包字节数
Input	Port index, Stream index
Output	Bytes number
Note	
Example	*pidx = self.get_portidx(0,2,1)* *sidx = 0* *counter_bytes = self.get_counter_stream_tx_pkts(pidx, sidx)*

10 Capture 相關

开启/关闭端口的撷取封包功能，当开启后，一旦端口有流量包，都会被端口撷取纪录，等待下 Stop 后，撷取的包陆续回传给 Parser 处理，整合存成 pcap 中介档，再经过 wireshark 原生工具-tshark 分析，最终以列表形式储存。用户可以选择观看撷取包的原生内容，以及撷取包的分析内容。

10.1 capture_frames_start(pidx, capture_type)

Function	特定端口开始 capture 包
Input	Port index, capture_type
Output	N/A
Note	nuconst.CAPTURE_ALL, capture 所有包 nuconst.CAPTURE_UNDERSIZE, capture undersize 包 nuconst.CAPTURE_OVERSIZE, capture oversize 包 nuconst.CAPTURE_JUMBO , capture jumbo 包
Example	*pidx = self.get_portidx(0,2,1)* *self.capture_frames_start(pidx, nuconst.CAPTURE_ALL)*

10.2 capture_frames_stop(pidx, capture_num)

Function	特定端口停止 capture 包
Input	Port index, capture_number
Output	N/A
Note	停止收包之后，同时将储存之包，送由后端分析结果。最终会产生几个中介档，pcap 为 capture 之实体包，xml 为分析 capture 包之后的信息
Example	*pidx = self.get_portidx(0,2,1)* *self.capture_frames_stop(pidx, 100)*

10.3 show_packet_content(pidx, fidx)

Function	显示特定端口，特定个包的内容
Input	Port index, frame index
Output	N/A
Note	
Example	*pidx = self.get_portidx(0,2,1)* *fidx = 1* *self.show_packet_content(pidx, fidx)*

10.4 show_packet_info(pidx, fidx)

Function	显示特定端口，特定个包的分析信息
Input	Port index, frame index
Output	N/A
Note	
Example	*pidx = self.get_portidx(0,2,1)* *fidx = 1* *self.show_packet_info(pidx, fidx)*

11 PING / ARP / DHCP 相關

本章节展示端口的网络第三层功能。包括端口自动回复 ARP Reply，当端口收到远端询问的 ARP Request 包时，此功能可以帮助询问端获端口取 MAC 等信息。端口执行 PING 功能，一方面探索被 PING 端口存在与否，另一方面可以藉此获得被 PING 端口的 TTL 等信息。再来是端口执行 DHCP Client 功能，DHCP Client 向 Server 取得一组可用 IP，得进行 4 次握手协定。藉由整合命令，最终成功的话可以获取 Server 端 IP 以及被分配到的 IP。

11.1 AutoARPReply 設置

此章节说明端口开启自动回覆 ARP Request 功能。每个端口可以设置 24 组配置(node)，分别有 24 组 MAC, VLAN, IPv4, IPv6。用户可以依据使用需求，开启不同的 ARP 配置，以 index 指名方式。被开启的配置，皆可以在有 ARP Request 时，自动回覆 ARP Reply 包。底下就配置与功能设置说明。

11.1.1 config_arp_enablenode(nodeidx, enable)

Function	以 index 方式开启/关闭 24 个配置其中之一。
Input	node index, enable
Output	N/A
Note	
Example	*Nodeidx = 0* *self.config_arp_enablenode(Nodeidx,True)* *self.arp_reply_start(port_idx)*

11.1.2 config_arp_mac(nodeidx, mymac)

Function	以 index 方式设置 mac
Input	node index, mac
Output	N/A
Note	
Example	*Nodeidx = 0* *mymac = “00:22:A2:00:04:01”* *self.config_arp_mac(Nodeidx, mymac)*

11.1.3 config_arp_vlan(nodeidx, myvlan)

Function	以 index 方式设置 VLAN
Input	node index, vlan

Output	N/A
Note	
Example	*Nodeidx = 0* *VlanID = "0x8100"* *self.config_arp_vlan(Nodeidx, VlanID)*

11.1.4 config_arp_ipv4(nodeidx, myip)

Function	以 index 方式设置 IPv4
Input	node index, ip
Output	N/A
Note	
Example	*Nodeidx = 0* *sip = "192.168.4.1"* *self.config_arp_ipv4(Nodeidx, sip)*

11.1.5 config_arp_gateway(nodeidx, gateway)

Function	以 index 方式设置 Gateway
Input	node index, gateway ip
Output	N/A
Note	
Example	*Nodeidx = 0* *gip = "192.168.4.254"* *self.config_arp_gateway(Nodeidx, gip)*

11.1.6 config_arp_ipv6(nodeidx, myipv6)

Function	以 index 方式设置 IPv6
Input	node index, ipv6
Output	N/A
Note	
Example	*Nodeidx = 0* *ipv6 = "FE80:0000:0000:0000:0000:0000:C0A8:0401"* *self.config_arp_ipv6(Nodeidx, ipv6)*

11.1.7 config_arp_gatewayv6(nodeidx, gatewayv6)

Function	以 index 方式设置 Gatewayv6
Input	node index, gateway ipv6

Output	N/A
Note	
Example	*Nodeidx = 0* *gipv6 = "FE80:0000:0000:0000:0000:0000:C0A8:04FF"* *self.config_arp_gatewayv6(Nodeidx, gipv6)*

11.1.8 arp_reply_start(portidx)

Function	加载 config 配置，并开始自动回覆 ARP Reply
Input	Port index
Output	N/A
Note	若是成功，板卡会回报信息包，NuPython 处理过后，将 ARP Reply 取得之 MAC，储存在 ns_info_portlist 中，目前没有取用端口特定信息的命令，故当作内部保存，供后续功能使用。
Example	*nscmd = NuPython.NuStreamsModuleSetting()* *Nodeidx = 0* *Portidx = 0* *mymac = "00:22:A2:00:04:01"* *myip = "192.168.4.2"* *gip = "192.168.4.254"* *nscmd.config_arp_enablenode(Nodeidx , True)* *nscmd.config_arp_mac(Nodeidx , mymac)* *nscmd.config_arp_ipv4(Nodeidx , myip)* *nscmd.config_arp_gateway(Nodeidx , gip)* *# Port start ARP* *nscmd.arp_reply_start(Portidx)*

11.2 PING 設置

11.2.1 config_ping_num_ping(num)

Function	设置端口發 ping 数量
Input	number
Output	N/A
Note	默認 number 為 4
Example	*number = 4* *self.config_ping_num_ping(number)*

11.2.2 config_ping_num_arp(num)

Function	设置端口發 ARP Request 數量
Input	number
Output	N/A
Note	默認 number 為 4
Example	*number = 4* *self.config_ping_num_arp(number)*

11.2.3 config_ping_num_ndp(num)

Function	设置端口發 NDP 數量
Input	number
Output	N/A
Note	默認 number 為 4
Example	*number = 4* *self.config_ping_num_ndp(number)*

11.2.4 config_ping_sip(ip)

Function	设置端口的 Source IP
Input	IPv4
Output	N/A
Note	
Example	*sip = "192.168.4.1"* *self.config_ping_sip(sip)*

11.2.5 config_ping_dip(ip)

Function	设置端口的 Destination IP
Input	IPv4
Output	N/A
Note	
Example	*dip = "192.168.4.2"* *self.config_ping_dip(dip)*

11.2.6 config_ping_gip(ip)

Function	设置端口的 Gateway IP

Input	IPv4
Output	N/A
Note	
Example	*gip = "192.168.4.254"* *self.config_ping_gip(gip)*

11.2.7 config_ping_smac(mac)

Function	設置端口的 MAC
Input	mac
Output	N/A
Note	
Example	*mac = "00:22:A2:00:04:01"* *self.config_ping_smac(mac)*

11.2.8 config_ping_sipv6(ipv6)

Function	设置端口的 Source IPv6
Input	IPv6
Output	N/A
Note	
Example	*sipv6 = "FE80:0000:0000:0000:0000:0000:C0A8:04FF"* *self.config_ping_sipv6(sipv6)*

11.2.9 config_ping_dipv6(ipv6)

Function	设置端口的 Destination IPv6
Input	IPv6
Output	N/A
Note	
Example	*dipv6 = "FE80:0000:0000:0000:0000:0000:C0A8:04FF"* *self.config_ping_dipv6(dipv6)*

11.2.10 config_ping_gipv6(ipv6)

Function	设置端口的 Gateway IPv6
Input	IPv6
Output	N/A
Note	
Example	*gipv6 = "FE80:0000:0000:0000:0000:0000:C0A8:04FF"*

	self.config_ping_gipv6(gipv6)

11.2.11 pingv4_send(pidx)

Function	加载 PING 配置，并开始自動執行 PING 程序
Input	Port index
Output	N/A
Note	若是成功，板卡会回报信息包，NuPython 处理过后，将自 ICMP Reply 取得之 IP 内容，除 TTL，Data length 等，储存在 ns_info_portlist 中，目前没有取用端口特定信息的命令，故当作内部保存，供后续功能使用。
Example	*nscmd = NuPython.NuStreamsModuleSetting()* *number = 4* *mac = "00:22:A2:00:04:01"* *ping_sip = "192.168.4.1"* *ping_dip = "192.168.4.2"* *ping_gip = "192.168.4.254"* *port_idx = 0* *nscmd.config_ping_num_ping(number)* *nscmd.config_ping_num_arp(number)* *nscmd.config_ping_sip(ping_sip)* *nscmd.config_ping_dip(ping_dip)* *nscmd.config_ping_gip(ping_gip)* *nscmd.config_ping_smac(mac)* *# Port Start Ping* *nscmd.pingv4_send(port_idx)*

11.2.12 pingv6_send(pidx)

Function	加载 PINGv6 配置，并开始自動執行 PINGv6 程序
Input	Port index
Output	N/A
Note	
Example	*nscmd = NuPython.NuStreamsModuleSetting()* *number = 4* *mac = "00:22:A2:00:04:01"* *ping_sipv6 = "FE80:0000:0000:0000:0000:0000:C0A8:0401"* *ping_dipv6 = "FE80:0000:0000:0000:0000:0000:C0A8:0402"* *ping_gipv6 = "FE80:0000:0000:0000:0000:0000:C0A8:04FF"* *port_idx = 0* *nscmd.config_ping_num_ping(number)*

	nscmd.config_ping_num_ndp(number) *nscmd.config_ping_sipv6(ping_sipv6)* *nscmd.config_ping_dipv6(ping_dipv6)* *nscmd.config_ping_gipv6(ping_gipv6)* *nscmd.config_ping_smac(mac)* *# Port Start Ping* *nscmd.pingv6_send(port_idx)*

11.3 DHCP 设置

11.3.1 config_dhcp_mac(mac)

Function	设置端口的 MAC
Input	Mac address
Output	N/A
Note	
Example	*mac = "00:22:A2:00:04:01"* *self.config_dhcp_mac(gip)*

11.3.2 dhcp_set(port_idx)

Function	端口的 DHCP 设置
Input	Port index
Output	N/A
Note	默认值: discover timeout=5000ms, ack timeout=5000ms, decline delay=1000ms, ack delay=1000ms， 目前这些参数不开放调整
Example	*nscmd = NuPython.NuStreamsModuleSetting()* *port_idx = 0* *mac = "00:22:A2:00:04:01"* *nscmd.config_dhcp_mac(mac)* *# start dhcp* *nscmd.dhcp_set(port_idx)*

11.3.3 dhcp_discovery(port_idx)

Function	启动端口的 DHCP discovery 程序
Input	Port index
Output	N/A
Note	若是成功，板卡会回报信息包，NuPython 处理过后，将自 DHCP Server 取得之 Client IP，Server IP 等信息，储存在 ns_info_portlist 中，目前没有取用端口特定信息的命令，故当作内部保存，供后续功能使用。
Example	*nscmd = NuPython.NuStreamsModuleSetting()* *port_idx = 0* *mac = "00:22:A2:00:04:01"* *nscmd.config_dhcp_mac(mac)* *# start dhcp* *nscmd.dhcp_set(port_idx)* *nscmd.dhcp_discovery(port_idx)*

12 NuPOE(T451)相关命令

NuPOE 板卡和一般 NuStreams 板卡最大不同之处，在于 Chassis ID。NuPOE 板卡的 Chassis ID 是以 MAC 为记录，因此需要 6 个位元组空间。其余使用和设置方式，都和 NuStreams 板卡雷同。而在使用上，NuPOE 机箱考量可串接，最多达 16 台，因此 NuPython 已经先将 NuPOE 机箱的 Chassis ID 以 List 方式建制好，用户只需要以 index 方式指明即可。每个 NuPOE 板卡，皆只有一个端口，因此不必设置端口(port)index，NuPython 默认为 1。

12.1 一般命令

一般命令指的是非测试相关的命令，比方读取板卡版本信息等等。

12.1.1 t451_server_connect(ip)

Function	和 NuPOE Server 以 IP 连线
Input	IP。String
Output	None
Note	
Example	*ip = 192.168.1.8* *self.t451_server_connect(ip)*

12.1.2 t451_server_disconnect()

Function	断开和 NuPOE Server 连线。和 connect 相对应。
Input	None
Output	None
Note	
Example	*self.t451_server_disconnect()*

12.1.3 t451_read_info_allport()

Function	读取所有板卡状态
Input	Chassis Index, Board Index
Output	None
Note	可以获取板卡 card type(目前搭配板卡皆为 t451)，lock 状态，版本号等等。 用户下了此命令之后，会一次回报 16 个槽的板卡状态。
Example	*self.t451_read_info_allport()*

12.1.4 t451_read_info_license_chassis(chassis_idx)

Function	读取机箱 license 状态
Input	Chassis Index
Output	None
Note	目前尚未实现处理 license 信息
Example	*self.t451_read_info_license_chassis(cidx)*

12.1.5 t451_read_info_license(chassis_idx, board_idx)

Function	读取个别板卡 license 状态
Input	Chassis Index, Board Index
Output	None
Note	目前尚未实现处理 license 信息
Example	*self.t451_read_info_license_chassis(cidx, bidx)*

12.2 Group 命令

将某些板卡设置为同一个 Group 之后，可以以 Group 方式下命令，省去一个个下命令的时间。

12.2.1 t451_port_mark(chassis_idx, board_idx)

Function	分别标记需要设置同一个 group 的端口，透过标记，配合 set_group 命令，即可设置为相同 group
Input	Chassis Index, Board Index
Output	None
Note	
Example	*self.t451_port_mark(cidx, bidx)*

12.2.2 t451_port_unmark(chassis_idx, board_idx)

Function	移除端口设置的标记。
Input	Chassis Index, Board Index
Output	None
Note	
Example	*self.t451_port_unmark(cidx, bidx)*

12.2.3 t451_set_group(group_id)

Function	将早先标记的端口设置为相同 group
Input	Group ID
Output	None
Note	
Example	*gid = 1* *cidx = 0* *slot1_idx = 0* *slot2_idx = 1* *self..t451_port_mark(cidx, slot1_idx)* *self.t451_port_mark(cidx, slot2_idx)* *self.t451_set_group(gid)*

12.2.4 t451_gopen_relay(group_id, is_open)

Function	将设置为同一个 group id 的端口开关继电器
Input	Group, On/Off
Output	None
Note	
Example	*isOpen = 1 # on* *self.t451_gopen_relay(gid, isOpen)* *isOpen = 0 # off* *self.t451_gopen_relay(gid, isOpen)*

12.2.5 t451_gstop_test(group_id)

Function	将设置为同一个 group id 的端口停止测试
Input	Group ID
Output	None
Note	
Example	*self.t451_gstop_test(gid)*

12.2.6 t451_gcounter_read_start(group_id, rate)

Function	将设置为同一个 group id 的端口开始读取 counter，rate 参数代表频率，单位是 Hz(每秒几次)
Input	Group ID, Rate
Output	None

Note	
Example	*sample_rate = 2 # 2 times in 1 sec.* *self.t451_gcounter_read_start(gid, sample_rate)*

12.2.7 t451_gcounter_read_stop(group_id)

Function	将设置为同一个 group id 的端口停止读取 counter
Input	Group ID
Output	None
Note	
Example	*self.t451_gcounter_read_stop(gid)*

12.2.8 t451_gcounter_clear(group_id)

Function	将设置为同一个 group id 的端口清除 counter
Input	Group ID
Output	None
Note	
Example	*self.t451_gcounter_clear(gid)*

12.3 控制命令

实际控制板卡动作的命令。执行控制命令之前，必须先下配置命令。配置命令之后，板卡会根据配置做出反应。任何控制板卡命令第一步，都需要先将板卡 lock，才能使用，否则 Server 不会有反应。

12.3.1 t451_open_relay(chassis_idx, board_idx, isopen)

Function	端口继电器开关
Input	Chassis Index, Board Index, On/Off
Output	None
Note	
Example	*isOpen = 1 # on* *cidx = 0* *slot1_idx = 0* *self.t451_gopen_relay(cidx, slot1_idx, isOpen)* *isOpen = 0 # off* *self.t451_gopen_relay(cidx, slot1_idx, isOpen)*

12.3.2 t451_port_lock(chassis_idx, board_idx, status)

Function	将要进行测试的端口保留，和 NuStreams 不同，NuStreams 可以以 map 方式设置多口同时，而 NuPOE 只能一个端口一个端口设置。
Input	Chassis Index, Board Index, Lock/UnLock
Output	None
Note	
Example	*Lockstatus = 1 # lock* *self.t451_port_lock(cidx, slot1_idx, lockstatus)*

12.3.3 t451_start_loading(chassis_idx, board_idx)

Function	依据配置，开始进行 loading 测试
Input	Chassis Index, Board Index
Output	None
Note	
Example	*self.t451_start_loading(cidx, slot1_idx)*

12.3.4 t451_stop_loading(chassis_idx, board_idx)

Function	停止 loading 测试
Input	Chassis Index, Board Index
Output	None
Note	
Example	*self.t451_stop_loading(cidx, slot1_idx)*

12.3.5 t451_start_sample(chassis_idx, board_idx)

Function	依据配置，开始进行采样
Input	Chassis Index, Board Index
Output	None
Note	
Example	*self.t451_start_sample(cidx, slot1_idx)*

12.3.6 t451_start_connect(chassis_idx, board_idx)

Function	依据配置，开始进行 connect 测试
Input	Chassis Index, Board Index
Output	None

Note	板卡正常情况，会在结束时回报结果状态，用户可在一段时间后读取结果
Example	*self.t451_start_connect(cidx, slot1_idx)*

12.3.7 t451_start_disconnect(chassis_idx, board_idx)

Function	依据配置，开始进行 disconnect 测试
Input	Chassis Index, Board Index
Output	None
Note	板卡正常情况，会在结束时回报结果状态，用户可在一段时间后读取结果
Example	*self.t451_start_disconnect(cidx, slot1_idx)*

12.3.8 t451_start_overload(chassis_idx, board_idx)

Function	依据配置，开始进行 overload 测试
Input	Chassis Index, Board Index
Output	None
Note	板卡正常情况，会在结束时回报结果状态，用户可在一段时间后读取结果
Example	*self.t451_start_overload(cidx, slot1_idx)*

12.3.9 t451_start_underload(chassis_idx, board_idx)

Function	依据配置，开始进行 underload 测试
Input	Chassis Index, Board Index
Output	None
Note	板卡正常情况，会在结束时回报结果状态，用户可在一段时间后读取结果
Example	*self.t451_start_underload(cidx, slot1_idx)*

12.3.10 t451_start_shorttest(chassis_idx, board_idx)

Function	依据配置，开始进行 short circuit 测试
Input	Chassis Index, Board Index
Output	None
Note	板卡正常情况，会在结束时回报结果状态，用户可在一段时间后读取结果
Example	*self.t451_start_shorttest(cidx, slot1_idx)*

12.3.11 t451_start_lldpload(chassis_idx, board_idx)

Function	依据配置，开始进行 LLDP loading 测试(未验证)
Input	Chassis Index, Board Index
Output	None

Note	板卡正常情况，会在结束时回报结果状态，用户可在一段时间后读取结果
Example	*self.t451_start_lldpload(cidx, slot1_idx)*

12.3.12 t451_stop_test(chassis_idx, board_idx)

Function	停止任何测试
Input	Chassis Index, Board Index
Output	None
Note	
Example	*self.t451_stop_test(cidx, slot1_idx)*

12.3.13 t451_counter_read_start(chassis_idx, board_idx, rate)

Function	端口开始读取 counter，rate 参数代表频率，单位是 Hz(每秒几次)
Input	Chassis Index, Board Index
Output	None
Note	
Example	*sample_rate = 2 # 2 times in 1 sec.* *self.t451_counter_read_start(cidx, slot1_idx, sample_rate)*

12.3.14 t451_counter_read_stop(chassis_idx, board_idx)

Function	端口停止读取 counter
Input	Chassis Index, Board Index
Output	None
Note	
Example	*self.t451_counter_read_stop(cidx, slot1_idx)*

12.3.15 t451_counter_read_once(chassis_idx, board_idx)

Function	读取特定端口的 counter 数值，一次性的。通常少用，会使用 counter_read_start，一次读回整个过程数据。
Input	Chassis Index, Board Index
Output	None
Note	
Example	*self.t451_counter_read_once(cidx, slot1_idx)*

12.3.16 t451_counter_clear(chassis_idx, board_idx)

Function	清除特定端口 counter 数值

Input	Chassis Index, Board Index
Output	None
Note	
Example	*self.t451_counter_clear(cidx, slot1_idx)*

12.4 配置相关命令

这章节主要设置所有测试相关的参数，保存在 NuPython 中，设置完成后，需使用 t451_set_test()函数，才能将数值吃进去。

12.4.1 t451_config_poeclass(val)

Function	设置 POE Class
Input	Value = 0~4
Output	None
Note	
Example	*poe_class = 0* *self.t451_config_poeclass(poe_class)*

12.4.2 t451_config_duttype(val)

Function	设置 DUT Type
Input	● T451_CFG_DUTTYPE_PSE ● T451_CFG_DUTTYPE_MIDSPAN
Output	None
Note	需引用 nuconst.py，或直接给值
Example	*import nuconst* *nsconst = nuconst.NuStreamsConst()* *self.t451_config_duttype(nsconst.T451_CFG_DUTTYPE_PSE)*

12.4.3 t451_config_alternative(val)

Function	设置极性
Input	● T451_CFG_ALTER_1236 ● T451_CFG_ALTER_4578 ● T451_CFG_ALTER_ALL
Output	None
Note	需引用 nuconst.py，或直接给值
Example	*import nuconst* *nsconst = nuconst.NuStreamsConst()*

	self.t451_config_duttype(nsconst.T451_CFG_ALTER_1236)

12.4.4 t451_config_cabletype(val)

Function	设置缆线形态
Input	● T451_CFG_CABLE_CAT3 ● T451_CFG_CABLE_CAT5 ● T451_CFG_CABLE_CAT7 ● T451_CFG_CABLE_CAT7A
Output	None
Note	需引用 nuconst.py，或直接给值
Example	*import nuconst* *nsconst = nuconst.NuStreamsConst()* *self.t451_config_cabletype(nsconst.T451_CFG_CABLE_CAT7)*

12.4.5 t451_config_cablelen(val)

Function	设置缆线长度
Input	Value = Length in meter
Output	None
Note	
Example	*self.t451_config_cablelen(10) #表示 10 米*

12.4.6 t451_config_copperloss(val)

Function	设置 T451 自身电阻损耗
Input	Default = 0, Value = 0~262144
Output	None
Note	Value = 0 表示忽略此参数，硬件自行设定
Example	*self.t451_config_copperloss(value)*

12.4.7 t451_config_poweralert(val)

Function	启动 Power 不正常时的 alert
Input	Value = 0~1
Output	None
Note	0:Disable, 1:Enable
Example	*self.t451_config_poweralert(0)*

12.4.8 t451_config_tempthreshold(val)

Function	设置温度阀值
Input	Value = 0~100
Output	None
Note	
Example	*self.t451_config_tempthreshold(poe_class)*

12.4.9 t451_config_tempalert(val)

Function	温度过高 alert
Input	Value = 0~1
Output	None
Note	0:Disable, 1:Enable
Example	*self.t451_config_tempalert(0)*

12.4.10 t451_config_reporttype(val)

Function	设置报告回报模式
Input	● T451_CFG_REPORT_NONE ● T451_CFG_REPORT_SAMPLE ● T451_CFG_REPORT_FINAL ● T451_CFG_REPORT_BOTH
Output	None
Note	T451_CFG_REPORT_NONE - 不回报任何报告 T451_CFG_REPORT_SAMPLE - 完成测试后回报 Sample 报告 T451_CFG_REPORT_FINAL - 完成测试后回报结果报告 T451_CFG_REPORT_BOTH - 完成测试后回报 Sample 和结果报告
Example	*import nuconst* *nsconst = nuconst.NuStreamsConst()* *self.t451_config_reporttype(nsconst.T451_CFG_REPORT_BOTH)*

12.4.11 t451_config_voltpoweron(val)

Function	设置 Power on 的电压
Input	Value。默认值=4800，单位为 0.01V
Output	None
Note	
Example	*self.t451_config_voltpoweron(4800)*

12.4.12 t451_config_voltpoweroff(val)

Function	设置 Power off 的电压
Input	Value。默认值=500，单位为 0.01V
Output	None
Note	
Example	*self.t451_config_voltpoweroff(500)*

12.4.13 t451_config_voltpowergood(val)

Function	设置 Power good 的电压
Input	Value。默认值=3600，单位为 0.01V
Output	None
Note	线上电压达到此值时，会触发 Power Good 信号。当处于非标准供电环境(非48V)时，必须设置此值。
Example	*self.t451_config_voltpowergood(3600)*

12.4.14 t451_config_voltpowerunder(val)

Function	设置 under power 数值
Input	Value。默认值=3700，单位为 0.01V
Output	None
Note	当数值低于此值时，判定为 under voltage
Example	*self.t451_config_voltpowerunder(3700)*

12.4.15 t451_config_voltpowertoohigh(val)

Function	设置 over power 数值
Input	Value。默认值=5700，单位为 0.01V
Output	None
Note	当数值高于此值时，判定为 over voltage
Example	*self.t451_config_voltpowertoohigh(5700)*

12.4.16 t451_config_conn_loadingflag(val)

Function	设置 Connect 测试时 Loading
Input	Value
Output	None
Note	Bit 0:320K Ohm

	Bit 1:10Uf Capacitor1 Bit 2:10Uf Capacitor2 Bit 3:0.44W Loading 默认全开(0xF)
Example	*self.t451_config_conn_loadingflag(0xF)*

12.4.17 t451_config_conn_timeout(val)

Function	设置 Connect 测试时连接超时时间
Input	Value。默认值=10000，单位为 ms。
Output	None
Note	
Example	*self.t451_config_conn_timeout(10000)*

12.4.18 t451_config_conn_waittime(val)

Function	设置 Connect 测试时连接后等待时间
Input	Value。默认值=1000，单位为 ms。
Output	None
Note	
Example	*self.t451_config_conn_waittime(1000)*

12.4.19 t451_config_over_power(val)

Function	设置过载测试的功率
Input	Value。默认值=5000，单位 0.01w。
Output	None
Note	
Example	*self.t451_config_over_power(5000)*

12.4.20 t451_config_over_timeout(val)

Function	设置过载测试的超时时间
Input	Value。默认值=3000，单位 ms。
Output	None
Note	
Example	*self.t451_config_over_timeout(3000)*

12.4.21 t451_config_under_power(val)

Function	设置 underload 测试的功率
Input	Value。默认值=1000，单位 0.01w。
Output	None
Note	
Example	*self.t451_config_under_power(1000)*

12.4.22 t451_config_under_timeout(val)

Function	设置 underload 测试的超时时间
Input	Value。默认值=3000，单位 ms。
Output	None
Note	
Example	*self.t451_config_under_timeout(poe_class)*

12.4.23 t451_config_short_timeout(val)

Function	设置 short circuit 测试的超时时间
Input	Value。默认值=3000，单位 ms。
Output	None
Note	
Example	*self.t451_config_short_timeout(3000)*

12.4.24 t451_config_disconn_timeout(val)

Function	设置断线测试的超时时间
Input	Value。默认值=3000，单位 ms。
Output	None
Note	
Example	*self.t451_config_disconn_timeout(3000)*

12.4.25 t451_config_load_mode(val)

Function	设置 Loading 测试的模式
Input	● T451_CFG_LOADING_FIX ● T451_CFG_LOADING_INC ● T451_CFG_LOADING_DEC ● T451_CFG_LOADING_RANDOM

	● T451_CFG_LOADING_STEP
Output	None
Note	● FIX：以 NormalLoadingPower 为固定负载功率测试 ● INC：从 MinLoadingPower 上升到 MaxLoadingPower ● DEC：从 MaxLoadingPower 下降到 MinLoadingPower ● RANDOM：在 MinLoadingPower 和 MaxLoadingPower 之间随机变化 ● STEP：按照后面表格方式变化
Example	*import nuconst* *nsconst = nuconst.NuStreamsConst()* *self.t451_config_load_mode(nsconst.T451_CFG_LOADING_STEP)*

12.4.26 t451_config_load_powermin(val)

Function	设置最小负载功率
Input	Value。默认值=1，单位 0.01w。
Output	None
Note	
Example	*self.t451_config_load_powermin(1) # 0.01w*

12.4.27 t451_config_load_powermax(val)

Function	设置最大负载功率
Input	Value。默认值=5000，单位 0.01w。
Output	None
Note	
Example	*self.t451_config_load_powermax(5000)*

12.4.28 t451_config_load_delay(index, val)

Function	loading 测试中，index 组的延迟时间
Input	Index, Value。Index 可添加 50 组，数值为 0~49。Value 默认为 3000，单位为 ms。
Output	None
Note	和 t451_config_load_normalpower 搭配
Example	*self.t451_config_load_delay(0,3000)*

12.4.29 t451_config_load_normalpower(index, val)

Function	loading 测试中，index 组的正常负载功率
Input	Index, Value。Index 可添加 50 组，数值为 0~49。Value 默认为 4000，单位为

	0.01w。
Output	None
Note	和 t451_config_load_delay 搭配
Example	*self.t451_config_laod_normalpower(0, 5000)*

12.4.30 t451_set_test(chassis_idx, board_idx)

Function	设置 POE 测试选项
Input	Chassis Index, Board Index
Output	None
Note	此命令会将前面相关设置值，设置到 T451 板卡，此时版卡尚不会动作，须搭配控制命令中的 start_xxx 命令才会依照设定值动作。亦即，在执行控制命令的测试命令前，务必先运行配置命令。
Example	*self.t451_config_set_test(cidx, slot1_idx)*

12.5 Report 命令

此章节命令，主要用于测试完成后，藉由命令显示结果。测试尚未完成或是测试失败，数值部份会呈现 0 值或是 Fail 结果为正常现象。

12.5.1 t451_report_connect(chassis_idx, board_idx)

Function	检视 Connect 测试报告
Input	Chassis Index, Board Index
Output	##### Connect Test Report ##### Power : No Power/Power Good Rise time : *num* us Inrush time : *num* us Inrush current : *num* mA Voltage 1 : *num* V Voltage 2 : *num* V Time 1 : *num* us Time 2 : *num* us Voltage class : *num* V Current : *num* mA PSE type : Unknown/AF/AT Inrush voltage high : *num* mV Inrush voltage low : *num* mV Poweron time : *num* ms Device type : PSE/Midspan

	Polarity : No Detect/Pair(12,36)/Pair(45,78)
Note	
Example	*self.t451_report_connect(cidx, slot1_idx)*

12.5.2 t451_report_disconnect(chassis_idx, board_idx)

Function	检视 Disconnect 测试报告
Input	Chassis Index, Board Index
Output	##### Disconnect Test Report #####
	Turnoff time : *num* ms
	Maintain time : *num* ms
	Voltage High : *num* mV
	Voltage low : *num* mV
	Result : Fail/Pass
Note	
Example	*self.t451_report_disconnect(cidx, slot1_idx)*

12.5.3 t451_report_overload(chassis_idx, board_idx)

Function	检视 Overload 测试报告
Input	Chassis Index, Board Index
Output	##### Overload Test Report #####
	Time : *num* ms
	Current : *num* mA
	Result : Fail/Pass
Note	
Example	*self.t451_report_overload(cidx, slot1_idx)*

12.5.4 t451_report_underload(chassis_idx, board_idx)

Function	检视 Underload 测试报告
Input	Chassis Index, Board Index
Output	##### Underload Test Report #####
	Time : *num* ms
	Current : *num* mA
	Result : Fail/Pass
Note	
Example	*self.t451_report_underload(cidx, slot1_idx)*

12.5.5 t451_report_shortcircuit(chassis_idx, board_idx)

Function	检视 Short Circuit 测试报告
Input	Chassis Index, Board Index
Output	##### Short Circuit Test Report #####
	Time : *num* ms
	Current in 10us : *num* mA
	Current in 50ms : *num* mA
	Result : Fail/Pass
Note	
Example	*self.t451_report_shortcircuit(cidx, slot1_idx)*

12.5.6 t451_report_loading(chassis_idx, board_idx)

Function	检视 Loading 测试报告
Input	Chassis Index, Board Index
Output	##### Loading Test Report #####
	Peak voltage : *num* V
	Min. voltage : *num* V
	Peak current : *num* mA
	Min. current : *num* mA
	Under voltage flag : N/A/Too Low
	Oer voltage flag : N/A/Too High
	Result : Fail/Pass
Note	多组 loading 报告尚未实现
Example	*self.t451_report_loading(cidx, slot1_idx)*

13 Callback 函數

回调函数和一般函数不同。在这个 NuPython API 中，所有函数都是被动呼叫，或是被动取得信息。而实际上的网络环境，以封包来说都是动态，实时的，如果是被动环境下去取得这些主动信息，势必得花上较多任务，而且不实时。要做到实时环境，仰赖能够主動接收 Server 端来的信息。回调函数就可以做到这点。

目前 NuPython 支援的回調函數，有一定格式：

```
def name_callback_func(cid, bid, pid, reportid)
# name_callback_func: 函数名，可以任意取
# cid : Chassis ID
# bid : Board ID
# pid : Port ID
# reported : Report ID，目前只支援 0x300-TxEnd report，0x400-LinkChange Report
```

回调函数简单来说可用三个步骤说明：

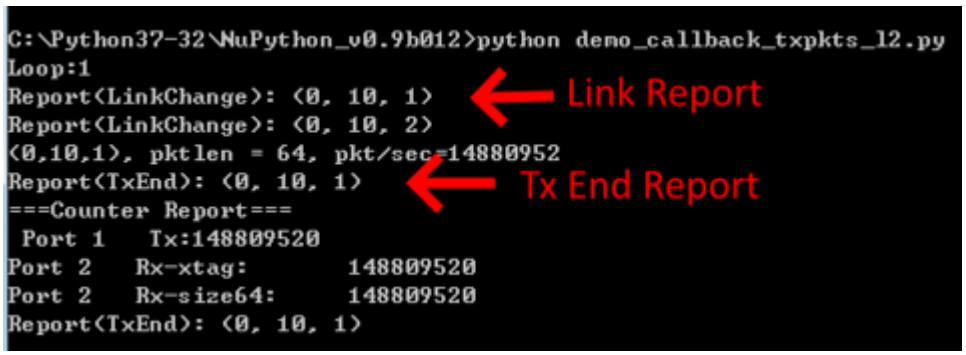
1. 客户端宣告回调函数: 用戶自行處理函數內部，範例只是顯示

```
def callback_function(cid, bid, pid, reportid):
    # user callback
    if reportid == 0x300:
        print(f"Report(TxEnd): ({cid}, {bid}, {pid})")
    elif reportid == 0x400:
        print(f"Report(LinkChange): ({cid}, {bid}, {pid})")
```

2. 將回调函数传入 Server 库

```
# 和傳統函數一樣，只是多了將剛剛宣告的回調函數代入
nscmd = NuPython.NuStreamsModuleSetting(callback_function)
```

3. Server 库需要时回调函数，Server 一接收到來自子卡的 Report 信息，就立即呼叫回調函數，如此一來，用戶剛剛的函數就會立即響應。



14 示例

本章节展示几个示例，更多示例在发行的 NuPython 底下 demo 档案夹。

13.1 获取板卡，端口信息

在屏幕上打印出 Module Information 以及 Media Status。

```
import time
import N趣uPython

n上scmd = NuPython.NuStreamsModuleSetting()

print 'Connect to Nuserver.....'
nscmd.server_connect('127.0.0.1')
print 'Wait for 3 seconds to Process Information Report.....'
time.sleep(3)
print '\n<---------- Module Information ---------->'
print 'Slot HWVer        FWVer        PROMVer     SN                    '
print '-------------------------------------------------------'
boardid = 1
print '%d       %s    %s    %s    %s' %(boardid, nscmd.get_version_hw(boardid), nscmd.get_version_fw(boardid),
nscmd.get_version_prom(boardid), nscmd.get_serialnum(boardid))
print ''
print 'Slot MAC               ManuDate              License       Date'
print '-------------------------------------------------------'
print '%d       %s %s %s %s' %(boardid, nscmd.get_macaddr(boardid), nscmd.get_manudate(boardid),
nscmd.get_license_mode(boardid), nscmd.get_license_date(boardid))

print '\n<---------- Module Information ---------->'
print 'Slot HWVer        FWVer        PROMVer     SN                    '
print '-------------------------------------------------------'
boardid = 4
print '%d       %s    %s    %s    %s' %(boardid, nscmd.get_version_hw(boardid), nscmd.get_version_fw(boardid),
nscmd.get_version_prom(boardid), nscmd.get_serialnum(boardid))
print ''
print 'Slot MAC               ManuDate              License       Date'
print '-------------------------------------------------------'
print '%d       %s %s %s %s' %(boardid, nscmd.get_macaddr(boardid), nscmd.get_manudate(boardid),
```

```
nscmd.get_license_mode(boardid), nscmd.get_license_date(boardid))


print '\n<---------- Media Type ---------->'
pidx = 0
while pidx < len(nscmd.ns_info_portlist):
    if nscmd.ns_info_portlist[pidx].boardID != 1 and nscmd.ns_info_portlist[pidx].boardID != 18:
        print '(%d,%d,%d) Speed:%s, Duplex:%s, AutoNegotiation:%s' %(nscmd.ns_info_portlist[pidx].chassisID,
nscmd.ns_info_portlist[pidx].boardID, nscmd.ns_info_portlist[pidx].portID, nscmd.get_media_speed(pidx),
nscmd.get_media_duplex(pidx), nscmd.get_media_autonego(pidx))
    pidx += 1
nscmd.port_unlock()
time.sleep(1)
nscmd.server_disconnect()
```

结果：

```
Connect to Nuserver......
Wait for 3 seconds to Process Information Report......

<---------- Module Information ---------->
Slot HWVer     FWVer     PROMVer   SN
-----------------------------------------------------
1    v0.9b002  v1.4b003  v1.6b002  0MNS600I0022

Slot MAC          ManuDate         License    Date
-----------------------------------------------------
1    0022a2020440 2012-01-01 00:00 Normal     2017/01

<---------- Module Information ---------->
Slot HWVer     FWVer     PROMVer   SN
-----------------------------------------------------
4    v4.1b003  v1.9b004  v1.6b011  0PRM78328847

Slot MAC          ManuDate         License    Date
-----------------------------------------------------
4    0022a206b520 2015-11-20 10:00 Evaluation 2016/12

<---------- Media Type ---------->
(0,4,1) Speed:1G, Duplex:Full, AutoNegotiation:Auto
(0,4,2) Speed:1G, Duplex:Full, AutoNegotiation:Auto
(0,4,3) Speed:LinkDown, Duplex:Full, AutoNegotiation:Auto
(0,4,4) Speed:LinkDown, Duplex:Full, AutoNegotiation:Auto
(0,7,1) Speed:LinkDown, Duplex:Full, AutoNegotiation:Auto
(0,7,2) Speed:LinkDown, Duplex:Full, AutoNegotiation:Auto
Press any key to continue . . .
```

13.2 收送包，获取端口 Stream 数据

为了简化呈现，不同的 port 用同样的参数。端口数据较 Stream 简单，故只呈现 Stream 数据。

```
import time
import NuPython
import nuconst
nscmd = NuPython.NuStreamsModuleSetting()
nsconst = nuconst.NuStreamsConst()
print 'Connect to Nuserver.....'
nscmd.server_connect('127.0.0.1')
print 'Wait for 3 seconds to Process Information Report.....'
time.sleep(3)

port1_idx = nscmd.get_portidx(0,4,1)
port2_idx = nscmd.get_portidx(0,4,2)
# lock port
nscmd.port_mark(0, 4, 1)
nscmd.port_mark(0, 4, 2)
nscmd.port_lock()

# transmit config. setting
nscmd.config_stream_pktlen(512)
# protocol
nscmd.config_stream_smac("00:22:A2:00:03:01")
nscmd.config_stream_dmac("00:22:A2:00:03:02")
nscmd.config_stream_utilization(100)
nscmd.config_stream_protocol(0)
# set total streams number
nscmd.config_stream_streamnum(2)
# set stream 1
nscmd.set_stream(port1_idx, 0)
nscmd.set_stream(port1_idx, 1)
# set stream 2
nscmd.set_stream(port2_idx, 0)
nscmd.set_stream(port2_idx, 1)

nscmd.config_tx_txpkts(1000)
nscmd.transmit_pkts(port1_idx)
nscmd.transmit_pkts(port2_idx)
```

```
time.sleep(2)
tx_p1_s1 = nscmd.get_counter_stream_tx_pkts(port1_idx, 0)
rx_p1_s1 = nscmd.get_counter_stream_rx_pkts(port1_idx, 0)
tx_p1_s2 = nscmd.get_counter_stream_tx_pkts(port1_idx, 1)
rx_p1_s2 = nscmd.get_counter_stream_rx_pkts(port1_idx, 1)

tx_p2_s1 = nscmd.get_counter_stream_tx_pkts(port2_idx, 0)
rx_p2_s1 = nscmd.get_counter_stream_rx_pkts(port2_idx, 0)
tx_p2_s2 = nscmd.get_counter_stream_tx_pkts(port2_idx, 1)
rx_p2_s2 = nscmd.get_counter_stream_rx_pkts(port2_idx, 1)

print "Port 1    Stream:1    Send:%i    Received:%i" % (tx_p1_s1, rx_p1_s1)
print "Port 1    Stream:2    Send:%i    Received:%i" % (tx_p1_s2, rx_p1_s2)
print "Port 2    Stream:1    Send:%i    Received:%i" % (tx_p2_s1, rx_p2_s1)
print "Port 2    Stream:2    Send:%i    Received:%i" % (tx_p2_s2, rx_p2_s2)
nscmd.port_unlock()
time.sleep(1)
nscmd.server_disconnect()
```

结果：

```
Connect to Nuserver......
Wait for 3 seconds to Process Information Report......
Port 1   Stream:1   Send:500   Received:500
Port 1   Stream:2   Send:500   Received:500
Port 2   Stream:1   Send:500   Received:500
Port 2   Stream:2   Send:500   Received:500
```

13.3 撷取包，并呈现包信息

撷取包之后，我们透过 wireshark 工具-tshark 工具，帮我们将包分析，因此可以看见输出的结果和 wireshark 非常雷同，信息的呈现上是非常正确的

```
from ctypes import *
import time,sys,traceback
import ctypes
import NuPython
import time
import nuconst
cid_1 = 0
bid_1 = 3
pid_1 = 1
```

```
cid_2 = 0
bid_2 = 3
pid_2 = 2

# packet size
pkt_len = 128
# packet count
pkt_count = 1000
# transmit rate
utilization = 10

##########################################
#    Standard running step
#    1. Connect to Server & Reserve Ports
#    2. Initial Counter
#    3. Config Tansmit Parameters
#    4. Start Capture Packets
#    5. Start Tansmit Packets
#    6. Wait for Tansmit Packets
#    7. Stop Capture Packets
#    8. Show Captured Packets Infomation
#    9. Release Ports and Disconnect
##########################################

# 1. Connect to Server & Reserve Ports
nscmd = NuPython.NuStreamsModuleSetting()
nsconst = nuconst.NuStreamsConst()
if (nscmd.server_connect("127.0.0.1")==0):
    print("Connect to server fail!")
# reserve ports
nscmd.port_mark(cid_1, bid_1, pid_1)
nscmd.port_mark(cid_2, bid_2, pid_2)
if (nscmd.port_lock() == 0):
    print("Lock port1 fail!")
    nscmd.server_disconnect()
port1_idx = nscmd.get_portidx(cid_1, bid_1, pid_1)
port2_idx = nscmd.get_portidx(cid_2, bid_2, pid_2)
# 2. Initial Counter
#    for correct speed
```

```
nscmd.read_info_link(port1_idx)
nscmd.read_info_link(port2_idx)
#    clear counter before setting
nscmd.clear_counter_port(port1_idx)
nscmd.clear_counter_port(port2_idx)
# 3. Config Tansmit Parameters
# mac address
mac_1 = "00:22:A2:%02X:%02X:%02X"%(cid_1, bid_1, pid_1)
mac_2 = "00:22:A2:%02X:%02X:%02X"%(cid_2, bid_2, pid_2)
# ip address
ip_1 = "192.168.%d.%d"%(bid_1, pid_1)
ip_2 = "192.168.%d.%d"%(bid_2, pid_2)
# start config
nscmd.config_stream_pktlen(pkt_len)
nscmd.config_stream_smac(mac_1)
nscmd.config_stream_dmac(mac_2)
nscmd.config_stream_sip(ip_1)
nscmd.config_stream_dip(ip_2)
nscmd.config_stream_utilization(utilization)
# protocol=1 means layer3, 2 means udp, 0 means layer2
nscmd.config_stream_protocol(1)
# single stream
nscmd.config_stream_streamnum(1)
nscmd.set_stream(port1_idx, 0)
nscmd.config_tx_txpkts(pkt_count)
# 4. Start Capture Packets - capture from port 2
nscmd.capture_frames_start(port2_idx, nsconst.CAPTURE_ALL)
# 5. Start Tansmit Packets - 0 means wait for a global transmit command
nscmd.config_tx_isimmediate(1)
nscmd.transmit_pkts(port1_idx)
# 6. Wait for Tansmit Packets
time.sleep(2)
# 7. Stop Capture Packets - capture packets immediately, so control stop time by self
nscmd.capture_frames_stop(port2_idx, 100)
time.sleep(1)
# 8. Show Captured Packets Infomation
# combine packets into a pcap file, call tshrak to process the pcap file to a xml file.
# then analysis xml file.
nscmd.show_packet_content(port2_idx, 1)
nscmd.show_packet_info(port2_idx, 1)
```

 # 9. Release Ports and Disconnect

nscmd.port_unlock()

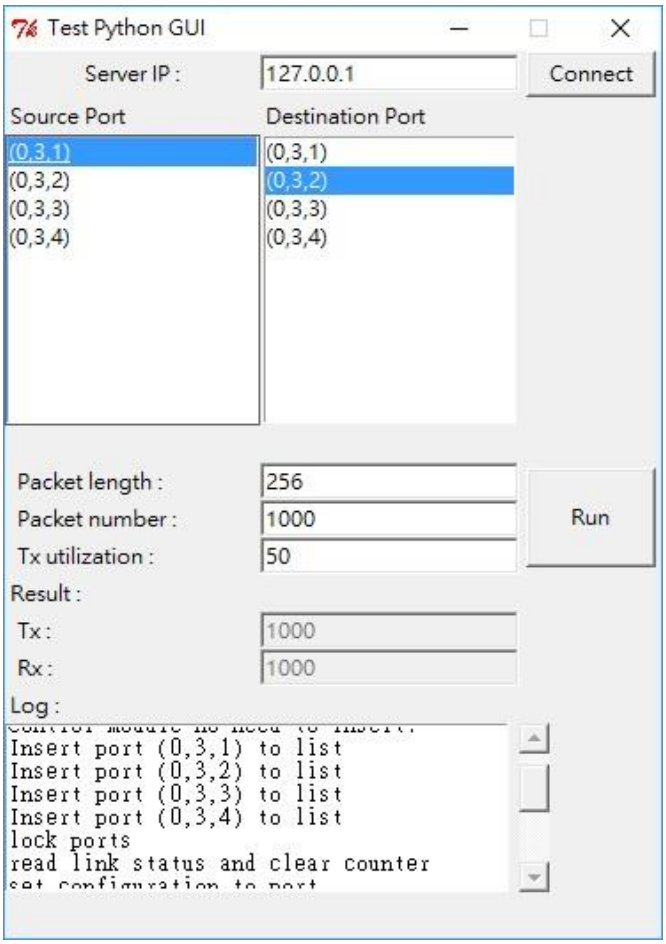
nscmd.server_disconnect()

結果:

```
C:\Python27>python D:\Project\20170516_PythonApplication1\PythonApplication1\demo\nupktstorage.py
Ethernet II, Src: Micro-St_df:99:98 (6c:62:6d:df:99:98), Dst: d8:fe:e3:c8:c2:7a (d8:fe:e3:c8:c2:7a)
    Destination: d8:fe:e3:c8:c2:7a (d8:fe:e3:c8:c2:7a)
    Address: d8:fe:e3:c8:c2:7a (d8:fe:e3:c8:c2:7a)
    .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Source: Micro-St_df:99:98 (6c:62:6d:df:99:98)
    Address: Micro-St_df:99:98 (6c:62:6d:df:99:98)
    .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IP (0x0800)
Internet Protocol Version 4, Src: 192.168.1.111 (192.168.1.111), Dst: 68.142.103.94 (68.142.103.94)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    0000 00.. = Differentiated Services Codepoint: Default (0x00)
    .... ..00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport) (0x00)
    Total Length: 40
    Identification: 0x4d1d (19741)
    Flags: 0x02 (Don't Fragment)
    0... .... = Reserved bit: Not set
    .1.. .... = Don't fragment: Set
    ..0. .... = More fragments: Not set
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (6)
    Header checksum: 0x3faf [correct]
    Good: True
    Bad: False
    Source: 192.168.1.111 (192.168.1.111)
    Source or Destination Address: 192.168.1.111 (192.168.1.111)
    Source Host: 192.168.1.111
    Source or Destination Host: 192.168.1.111
    Destination: 68.142.103.94 (68.142.103.94)
    Source or Destination Address: 68.142.103.94 (68.142.103.94)
    Destination Host: 68.142.103.94
    Source or Destination Host: 68.142.103.94
Transmission Control Protocol, Src Port: 58940 (58940), Dst Port: http (80), Seq: 1, Ack: 1, Len: 0
    Source port: 58940 (58940)
    Destination port: http (80)
    Source or Destination Port: 58940
    Source or Destination Port: 80
    Stream index: 0
    TCP Segment Len: 0
    Sequence number: 1    (relative sequence number)
    Acknowledgment number: 1    (relative ack number)
    Header length: 20 bytes
    Flags: 0x011 (FIN, ACK)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Nonce: Not set
    .... 0... .... = Congestion Window Reduced (CWR): Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...1 .... = Acknowledgment: Set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
    .... .... ..0. = Syn: Not set
    .... .... ...1 = Fin: Set
    Expert Info (Chat/Sequence): Connection finish (FIN)
    Message: Connection finish (FIN)
    Severity level: Chat
    Group: Sequence
    Window size value: 32708
    Calculated window size: 32708
    Window size scaling factor: -1 (unknown)
    Checksum: 0x6747 [validation disabled]
    Good Checksum: False
    Bad Checksum: False
```

13.4 以 UI 方式呈现收送包

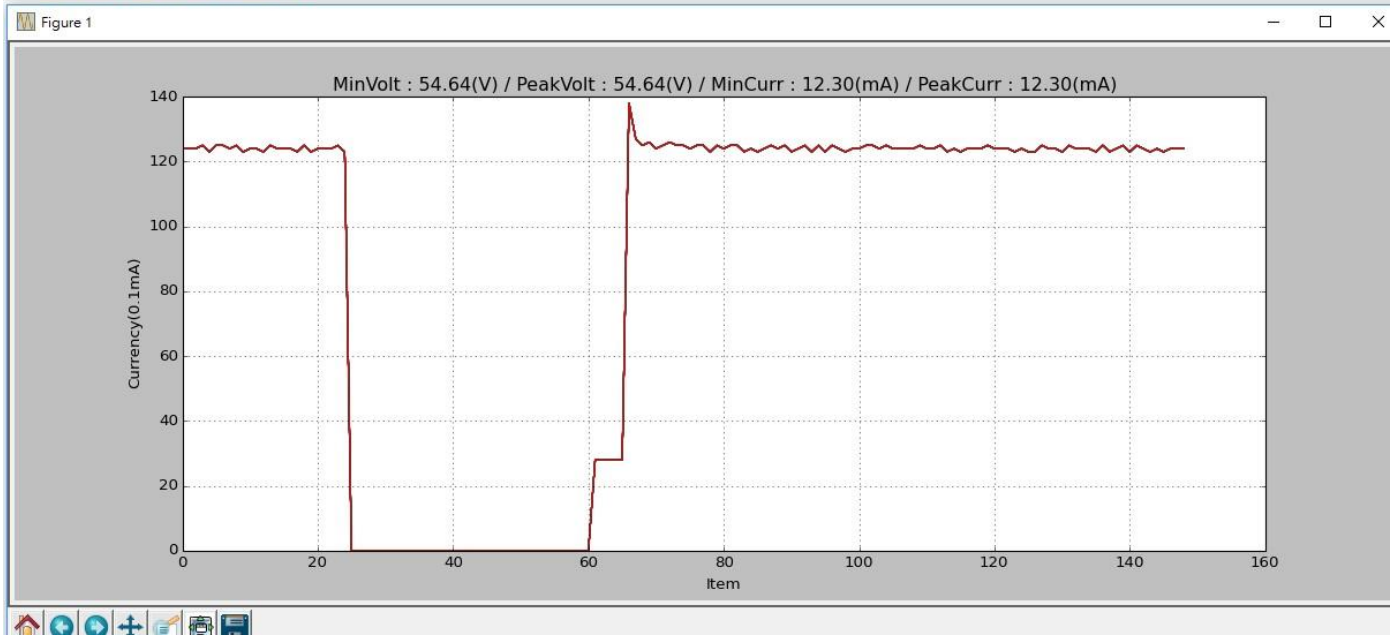
利用 Python 内建的套件 - Tkinter(Python2.7)。來呈現 UI。
結果:



13.5 T451 执行 connect 测试

除了显示最终结果之外，利用 Python 第三方套件 - matplotlib.pyplot，画出测试过程中的要电状态。范本挡案在发布文挡中「demo\NuPOE」挡案夹底下可参考。

```
    #print("(T451 Report) counter")
```




```
命令提示字元 - cmd - python  demo_t451_test_connect.py
    canvas.start_event_loop(interval)
  File "C:\Python27\lib\site-packages\matplotlib\backends\backend_tkagg.py", line 502, in start_eve
nt_loop
    FigureCanvasBase.start_event_loop_default(self,timeout)
  File "C:\Python27\lib\site-packages\matplotlib\backend_bases.py", line 2415, in start_event_loop_
default
    self.flush_events()
  File "C:\Python27\lib\site-packages\matplotlib\backends\backend_tkagg.py", line 499, in flush_eve
nts
    self._master.update()
  File "C:\Python27\lib\lib-tk\Tkinter.py", line 1019, in update
    self.tk.call('update')
_tkinter.TclError: can't invoke "update" command:  application has been destroyed

D:\Project\20170923_PythonApplication1\Release\demo\NuPOE>python demo_t451_test_connect.py
Connect to NuPOE Server.
Lock ports.
Set test config.
Start connect test-1.
Waiting for 10 sec.
(T451 Report) connect
Stop test.
Show connect test report...
#####  Connect Test Report  #####
  Power : Power Good
  Rise time : 5 us
  Inrush time : 4 us
  Inrush current : 1185.1 mA
  Voltage 1 : 5.8 V
  Voltage 2 : 4.1 V
  Time 1 : 65 us
  Time 2 : 248 us
  Voltage class : 17.94 V
  Current : 0.7 mA
  PSE type : AF
  Inrush voltage high : 56170 mV
  Inrush voltage low : 47720 mV
  Poweron time : 274 ms
  Device type : PSE
  Polarity : No Detect
Unlock port.
Disconnect.
C:\Python27\lib\site-packages\matplotlib\backend_bases.py:2407: MatplotlibDeprecationWarning: Using
 default event loop until function specific to this GUI is implemented
  warnings.warn(str, mplDeprecation)
```

```
from ctypes import *
import time,sys,traceback
import ctypes
import NuPython
import time
```

```
import nuconst
import matplotlib.pyplot as plt #  第三方套件
import numpy as np
import thread

nscmd = NuPython.NuStreamsModuleSetting()
nsconst = nuconst.NuStreamsConst()

class T451ConnectTest:
    def __init__(self):
        self.chassis_idx = 0
        self.slot1_idx = 0
        self.slot2_idx = 1
        self.groupid = 1
        self.lockstatus = 0
        self.unlockstatus = 1
        self.sample_rate = 2 # Hz
        self.isShowChart = False

    def ShowInstantVoltage(self, cidx, sidx):
        if self.sample_rate <= 10:
            plt.ion() ## Note this correction
            plt.xlabel('Item')
            plt.ylabel('Currency(0.1mA)')
            plt.grid(True)
            # for Line Chart
            while self.isShowChart == True:
                plt.title('MinVolt : %.2f(V) / PeakVolt : %.2f(V) / MinCurr : %.2f(mA) / PeakCurr : %.2f(mA)'
%(nscmd.t451_info_board[cidx].t451_board[sidx].report.counter_volt_min*0.01,
nscmd.t451_info_board[cidx].t451_board[sidx].report.counter_volt_peak*0.01,nscmd.t451_info_board[cidx].t451_board[sidx].report.coun
ter_curr_min*0.1, nscmd.t451_info_board[cidx].t451_board[sidx].report.counter_curr_peak*0.1))
                t = np.arange(0.0, nscmd.t451_info_board[cidx].t451_board[sidx].report.counter_idx, 1) # range 0~100, step = 1
                tmplist =
nscmd.t451_info_board[cidx].t451_board[sidx].report.counter_inst_curr[0:nscmd.t451_info_board[cidx].t451_board[sidx].report.counter_i
dx]
                plt.plot(t, tmplist)
                plt.show()
                plt.pause(0.5)
            plt.close()
```

```
def StartTest(self):
    print("Connect to NuPOE Server.")
    if (nscmd.t451_server_connect("192.168.1.8")==0): # connect to server
        print("Connect to server fail!")

    # lock ports per slot
    print("Lock ports.")
    if (nscmd.t451_port_lock(self.chassis_idx, self.slot1_idx, self.lockstatus) == 0):
        print("Lock port1 fail!")
        nscmd.t451_server_disconnect()

    # group mark
    nscmd.t451_port_mark(self.chassis_idx, self.slot1_idx)
    nscmd.t451_set_group(0) # set all port to group 0
    nscmd.t451_set_group(self.groupid) # set all port to group 1
    time.sleep(1)
    # Set test config
    # General config.
    print("Set test config.")
    nscmd.t451_config_poeclass(0)
    nscmd.t451_config_duttype(nsconst.T451_CFG_DUTTYPE_PSE)
    nscmd.t451_config_alternative(nsconst.T451_CFG_ALTER_1236)
    nscmd.t451_config_cabletype(nsconst.T451_CFG_CABLE_CAT5)
    nscmd.t451_config_cablelen(1)
    nscmd.t451_config_copperloss(0)
    nscmd.t451_config_poweralert(nsconst.ENABLE)
    nscmd.t451_config_tempthreshold(70) # temperature too high
    nscmd.t451_config_tempalert(nsconst.ENABLE)
    nscmd.t451_config_reporttype(nsconst.T451_CFG_REPORT_BOTH)
    nscmd.t451_config_voltpoweron(4800) # 0.01v
    nscmd.t451_config_voltpoweroff(500)
    nscmd.t451_config_voltpowergood(3600)
    nscmd.t451_config_voltpowerunder(3500)
    nscmd.t451_config_voltpowertoohigh(5700)
    # Connect config.
    nscmd.t451_config_conn_loadingflag(0xf)
    nscmd.t451_config_conn_timeout(10000) # ms
    nscmd.t451_config_conn_waittime(1000) # ms
    # Set Config.
    nscmd.t451_set_test(self.chassis_idx, self.slot1_idx)
```

```
        time.sleep(1)
        nscmd.t451_gcounter_read_start(self.groupid, self.sample_rate) # start counter - group
        time.sleep(1)
        # relay off per slot
        print("Start connect test-1.")
        nscmd.t451_open_relay(self.chassis_idx, self.slot1_idx, 1)
        time.sleep(1)
        self.isShowChart = True
        # start connect test
        nscmd.t451_start_connect(self.chassis_idx, self.slot1_idx)
        print("Waiting for 10 sec.")
        time.sleep(10)
        #self.isShowChart = False
        # relay off per slot
        nscmd.t451_open_relay(self.chassis_idx, self.slot1_idx, 0)
        time.sleep(1)

        # stop counter - group
        nscmd.t451_gcounter_read_stop(self.groupid)
        # stop test - group
        print("Stop test.")
        nscmd.t451_gstop_test(self.groupid)
        print("Show connect test report...")
        nscmd.t451_report_connect(self.chassis_idx, self.slot1_idx)
        # relay off - group
        nscmd.t451_gopen_relay(self.groupid, 0)
        time.sleep(1)

        print("Unlock port.")
        if (nscmd.t451_port_lock(self.chassis_idx, self.slot1_idx, self.unlockstatus) == 0):
            print("UnLock port1 fail!")
            nscmd.t451_server_disconnect()

        time.sleep(1)
        print("Disconnect.")
        nscmd.t451_server_disconnect()
        self.ShowInstantVoltage(self.chassis_idx, self.slot1_idx)

t451test = T451ConnectTest()
t451test.StartTest()
```

15 缺漏(未来版本补齐)

- 组合式命令，如 RouterNAT Test
- Serial 輸出
- Web Camera 接口