NuPython

(Python API

for NuStreams Systems)

Programming Guide

**Revision History**

Version : 1.6

Date : 2023/11/27

Author : Taken, Shih

|  |  |  |
| --- | --- | --- |
| **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 12](#_Toc151975403)

[2. Programming 13](#_Toc151975404)

[2.1 NuPython结构与规范 13](#_Toc151975405)

[2.2 函数命名规则 15](#_Toc151975406)

[2.3 编程步骤 16](#_Toc151975407)

[2.4 引用到的函数 18](#_Toc151975408)

[2.4.1 Python内建函数 18](#_Toc151975409)

[2.4.2 外部函数及程序 18](#_Toc151975410)

[3. 一般命令 19](#_Toc151975411)

[3.1 server\_connect(ip) 19](#_Toc151975412)

[3.2 server\_disconnect() 19](#_Toc151975413)

[3.3 port\_mark(chassis, board, port) / port\_unmark(chassis, board, port) 19](#_Toc151975414)

[3.4 port\_lock() / port\_unlock() 20](#_Toc151975415)

[3.5 get\_portidx(chassis, board, port) 20](#_Toc151975416)

[4 读取信息命令 21](#_Toc151975417)

[4.1 read\_info\_board(pidx) 21](#_Toc151975418)

[4.2 read\_license\_board(pidx) 21](#_Toc151975419)

[4.3 read\_info\_link(pidx) 21](#_Toc151975420)

[4.4 read\_counter\_port\_once(pidx) 22](#_Toc151975421)

[4.5 read\_counter\_port\_start(pidx, interval) 22](#_Toc151975422)

[4.6 read\_counter\_port\_stop(pidx) 22](#_Toc151975423)

[4.7 read\_counter\_stream\_once(pidx) 22](#_Toc151975424)

[4.8 read\_counter\_stream\_start(pidx, interval) 23](#_Toc151975425)

[5 切换Media命令 24](#_Toc151975426)

[5.1 config\_media\_speed(speed) 24](#_Toc151975427)

[5.2 config\_media\_duplex(duplex) 24](#_Toc151975428)

[5.3 config\_media\_autonego(negotiation) 24](#_Toc151975429)

[5.4 config\_media\_signal(signal) 25](#_Toc151975430)

[5.5 set\_media(pidx) 25](#_Toc151975431)

[6 数据清除，停止命令 26](#_Toc151975432)

[6.1 clear\_counter\_port(pidx) 26](#_Toc151975433)

[6.2 clear\_counter\_stream(pidx) 26](#_Toc151975434)

[6.3 read\_counter\_stop() 26](#_Toc151975435)

[7 配置端口相关命令 27](#_Toc151975436)

[7.1 config\_stream\_streamnum(number) 27](#_Toc151975437)

[7.2 config\_stream\_utilization(loading) 27](#_Toc151975438)

[7.3 config\_stream\_enable\_randomlen(enable) 27](#_Toc151975439)

[7.4 config\_stream\_pktlen(length) 27](#_Toc151975440)

[7.5 config\_stream\_streamid(sid) 28](#_Toc151975441)

[7.6 config\_stream\_enable\_vlan(enable) 28](#_Toc151975442)

[7.7 config\_stream\_vlan\_id(vid) 28](#_Toc151975443)

[7.8 config\_stream\_vlan\_pri(priority) 28](#_Toc151975444)

[7.9 config\_stream\_protocol(protocol\_id) 29](#_Toc151975445)

[7.10 config\_stream\_smac(mac) 29](#_Toc151975446)

[7.11 config\_stream\_dmac(mac) 29](#_Toc151975447)

[7.12 config\_stream\_arp\_smac(mac) 29](#_Toc151975448)

[7.13 config\_stream\_arp\_dmac(mac) 30](#_Toc151975449)

[7.14 config\_stream\_arp\_sip(ip) 30](#_Toc151975450)

[7.15 config\_stream\_arp\_dip(ip) 30](#_Toc151975451)

[7.16 config\_stream\_sip(ip) 30](#_Toc151975452)

[7.17 config\_stream\_dip(ip) 31](#_Toc151975453)

[7.18 config\_stream\_sport(port\_num) 31](#_Toc151975454)

[7.19 config\_stream\_dport(port\_num) 31](#_Toc151975455)

[7.20 config\_stream\_adderror (errorcode) 31](#_Toc151975456)

[7.21 config\_stream\_enable\_xtag (enable) 32](#_Toc151975457)

[7.22 set\_stream(pidx, sidx) 32](#_Toc151975458)

[8 传送相关命令 33](#_Toc151975459)

[8.1 config\_tx\_txtime(seconds) 33](#_Toc151975460)

[8.2 config\_tx\_txpkts(packets) 33](#_Toc151975461)

[8.3 config\_tx\_isimmediate(enable) 33](#_Toc151975462)

[8.4 config\_port\_flowctrl\_tx(enable) 33](#_Toc151975463)

[8.5 config\_port\_flowctrl\_rx(enable) 34](#_Toc151975464)

[8.6 set\_config\_rxstream(pidx) 34](#_Toc151975465)

[8.7 transmit\_pkts(pidx) 34](#_Toc151975466)

[8.8 transmit\_pkts\_stop(pidx) 34](#_Toc151975467)

[8.9 transmit\_pkts\_sync() 35](#_Toc151975468)

[8.10 transmit\_pkts\_sync\_stop() 35](#_Toc151975469)

[9 结果数值命令 35](#_Toc151975470)

[9.1 Module相关 35](#_Toc151975471)

[9.1.1 get\_version\_hw(slotid) 35](#_Toc151975472)

[9.1.2 get\_version\_fw(slotid) 36](#_Toc151975473)

[9.1.3 get\_version\_prom(slotid) 36](#_Toc151975474)

[9.1.4 get\_serialnum(slotid) 36](#_Toc151975475)

[9.1.5 get\_macaddr(slotid) 36](#_Toc151975476)

[9.1.6 get\_manudate(slotid) 37](#_Toc151975477)

[9.1.7 get\_license\_mode(slotid) 37](#_Toc151975478)

[9.1.8 get\_license\_date(slotid) 37](#_Toc151975479)

[9.1.9 get\_modelname(slotid) 37](#_Toc151975480)

[9.2 Link相关 38](#_Toc151975481)

[9.2.1 get\_media\_speed(pidx) 38](#_Toc151975482)

[9.2.2 get\_media\_duplex(pidx) 38](#_Toc151975483)

[9.2.3 get\_media\_autonego(pidx) 38](#_Toc151975484)

[9.2.4 get\_media\_signal(pidx) 39](#_Toc151975485)

[9.3 Port Counter相关 39](#_Toc151975486)

[9.3.1 get\_counter\_port(pidx, countidx) 39](#_Toc151975487)

[9.4 Stream Counter相关 40](#_Toc151975488)

[9.4.1 get\_counter\_stream\_rx\_pkts(pidx, sidx) 40](#_Toc151975489)

[9.4.2 get\_counter\_stream\_rx\_bytes(pidx, sidx) 41](#_Toc151975490)

[9.4.3 get\_counter\_stream\_rx\_latency(pidx, sidx) 41](#_Toc151975491)

[9.4.4 get\_counter\_stream\_tx\_pkts(pidx, sidx) 41](#_Toc151975492)

[9.4.5 get\_counter\_stream\_tx\_bytes(pidx, sidx) 41](#_Toc151975493)

[10 Capture相關 43](#_Toc151975494)

[10.1 capture\_frames\_start(pidx, capture\_type) 43](#_Toc151975495)

[10.2 capture\_frames\_stop(pidx, capture\_num) 43](#_Toc151975496)

[10.3 show\_packet\_content(pidx, fidx) 43](#_Toc151975497)

[10.4 show\_packet\_info(pidx, fidx) 44](#_Toc151975498)

[11 PING / ARP / DHCP相關 45](#_Toc151975499)

[11.1 AutoARPReply設置 45](#_Toc151975500)

[11.1.1 config\_arp\_enablenode(nodeidx, enable) 45](#_Toc151975501)

[11.1.2 config\_arp\_mac(nodeidx, mymac) 45](#_Toc151975502)

[11.1.3 config\_arp\_vlan(nodeidx, myvlan) 45](#_Toc151975503)

[11.1.4 config\_arp\_ipv4(nodeidx, myip) 46](#_Toc151975504)

[11.1.5 config\_arp\_gateway(nodeidx, gateway) 46](#_Toc151975505)

[11.1.6 config\_arp\_ipv6(nodeidx, myipv6) 46](#_Toc151975506)

[11.1.7 config\_arp\_gatewayv6(nodeidx, gatewayv6) 46](#_Toc151975507)

[11.1.8 arp\_reply\_start(portidx) 47](#_Toc151975508)

[11.2 PING設置 47](#_Toc151975509)

[11.2.1 config\_ping\_num\_ping(num) 47](#_Toc151975510)

[11.2.2 config\_ping\_num\_arp(num) 48](#_Toc151975511)

[11.2.3 config\_ping\_num\_ndp(num) 48](#_Toc151975512)

[11.2.4 config\_ping\_sip(ip) 48](#_Toc151975513)

[11.2.5 config\_ping\_dip(ip) 48](#_Toc151975514)

[11.2.6 config\_ping\_gip(ip) 48](#_Toc151975515)

[11.2.7 config\_ping\_smac(mac) 49](#_Toc151975516)

[11.2.8 config\_ping\_sipv6(ipv6) 49](#_Toc151975517)

[11.2.9 config\_ping\_dipv6(ipv6) 49](#_Toc151975518)

[11.2.10 config\_ping\_gipv6(ipv6) 49](#_Toc151975519)

[11.2.11 pingv4\_send(pidx) 50](#_Toc151975520)

[11.2.12 pingv6\_send(pidx) 50](#_Toc151975521)

[11.3 DHCP设置 51](#_Toc151975522)

[11.3.1 config\_dhcp\_mac(mac) 51](#_Toc151975523)

[11.3.2 dhcp\_set(port\_idx) 51](#_Toc151975524)

[11.3.3 dhcp\_discovery(port\_idx) 52](#_Toc151975525)

[12 NuPOE(T451)相关命令 53](#_Toc151975526)

[12.1 一般命令 53](#_Toc151975527)

[12.1.1 t451\_server\_connect(ip) 53](#_Toc151975528)

[12.1.2 t451\_server\_disconnect() 53](#_Toc151975529)

[12.1.3 t451\_read\_info\_allport() 53](#_Toc151975530)

[12.1.4 t451\_read\_info\_license\_chassis(chassis\_idx) 54](#_Toc151975531)

[12.1.5 t451\_read\_info\_license(chassis\_idx, board\_idx) 54](#_Toc151975532)

[12.2 Group命令 54](#_Toc151975533)

[12.2.1 t451\_port\_mark(chassis\_idx, board\_idx) 54](#_Toc151975534)

[12.2.2 t451\_port\_unmark(chassis\_idx, board\_idx) 54](#_Toc151975535)

[12.2.3 t451\_set\_group(group\_id) 55](#_Toc151975536)

[12.2.4 t451\_gopen\_relay(group\_id, is\_open) 55](#_Toc151975537)

[12.2.5 t451\_gstop\_test(group\_id) 55](#_Toc151975538)

[12.2.6 t451\_gcounter\_read\_start(group\_id, rate) 55](#_Toc151975539)

[12.2.7 t451\_gcounter\_read\_stop(group\_id) 56](#_Toc151975540)

[12.2.8 t451\_gcounter\_clear(group\_id) 56](#_Toc151975541)

[12.3 控制命令 56](#_Toc151975542)

[12.3.1 t451\_open\_relay(chassis\_idx, board\_idx, isopen) 56](#_Toc151975543)

[12.3.2 t451\_port\_lock(chassis\_idx, board\_idx, status) 57](#_Toc151975544)

[12.3.3 t451\_start\_loading(chassis\_idx, board\_idx) 57](#_Toc151975545)

[12.3.4 t451\_stop\_loading(chassis\_idx, board\_idx) 57](#_Toc151975546)

[12.3.5 t451\_start\_sample(chassis\_idx, board\_idx) 57](#_Toc151975547)

[12.3.6 t451\_start\_connect(chassis\_idx, board\_idx) 57](#_Toc151975548)

[12.3.7 t451\_start\_disconnect(chassis\_idx, board\_idx) 58](#_Toc151975549)

[12.3.8 t451\_start\_overload(chassis\_idx, board\_idx) 58](#_Toc151975550)

[12.3.9 t451\_start\_underload(chassis\_idx, board\_idx) 58](#_Toc151975551)

[12.3.10 t451\_start\_shorttest(chassis\_idx, board\_idx) 58](#_Toc151975552)

[12.3.11 t451\_start\_lldpload(chassis\_idx, board\_idx) 58](#_Toc151975553)

[12.3.12 t451\_stop\_test(chassis\_idx, board\_idx) 59](#_Toc151975554)

[12.3.13 t451\_counter\_read\_start(chassis\_idx, board\_idx, rate) 59](#_Toc151975555)

[12.3.14 t451\_counter\_read\_stop(chassis\_idx, board\_idx) 59](#_Toc151975556)

[12.3.15 t451\_counter\_read\_once(chassis\_idx, board\_idx) 59](#_Toc151975557)

[12.3.16 t451\_counter\_clear(chassis\_idx, board\_idx) 59](#_Toc151975558)

[12.4 配置相关命令 60](#_Toc151975559)

[12.4.1 t451\_config\_poeclass(val) 60](#_Toc151975560)

[12.4.2 t451\_config\_duttype(val) 60](#_Toc151975561)

[12.4.3 t451\_config\_alternative(val) 60](#_Toc151975562)

[12.4.4 t451\_config\_cabletype(val) 61](#_Toc151975563)

[12.4.5 t451\_config\_cablelen(val) 61](#_Toc151975564)

[12.4.6 t451\_config\_copperloss(val) 61](#_Toc151975565)

[12.4.7 t451\_config\_poweralert(val) 61](#_Toc151975566)

[12.4.8 t451\_config\_tempthreshold(val) 62](#_Toc151975567)

[12.4.9 t451\_config\_tempalert(val) 62](#_Toc151975568)

[12.4.10 t451\_config\_reporttype(val) 62](#_Toc151975569)

[12.4.11 t451\_config\_voltpoweron(val) 62](#_Toc151975570)

[12.4.12 t451\_config\_voltpoweroff(val) 63](#_Toc151975571)

[12.4.13 t451\_config\_voltpowergood(val) 63](#_Toc151975572)

[12.4.14 t451\_config\_voltpowerunder(val) 63](#_Toc151975573)

[12.4.15 t451\_config\_voltpowertoohigh(val) 63](#_Toc151975574)

[12.4.16 t451\_config\_conn\_loadingflag(val) 63](#_Toc151975575)

[12.4.17 t451\_config\_conn\_timeout(val) 64](#_Toc151975576)

[12.4.18 t451\_config\_conn\_waittime(val) 64](#_Toc151975577)

[12.4.19 t451\_config\_over\_power(val) 64](#_Toc151975578)

[12.4.20 t451\_config\_over\_timeout(val) 64](#_Toc151975579)

[12.4.21 t451\_config\_under\_power(val) 65](#_Toc151975580)

[12.4.22 t451\_config\_under\_timeout(val) 65](#_Toc151975581)

[12.4.23 t451\_config\_short\_timeout(val) 65](#_Toc151975582)

[12.4.24 t451\_config\_disconn\_timeout(val) 65](#_Toc151975583)

[12.4.25 t451\_config\_load\_mode(val) 65](#_Toc151975584)

[12.4.26 t451\_config\_load\_powermin(val) 66](#_Toc151975585)

[12.4.27 t451\_config\_load\_powermax(val) 66](#_Toc151975586)

[12.4.28 t451\_config\_load\_delay(index, val) 66](#_Toc151975587)

[12.4.29 t451\_config\_load\_normalpower(index, val) 66](#_Toc151975588)

[12.4.30 t451\_set\_test(chassis\_idx, board\_idx) 67](#_Toc151975589)

[12.5 Report命令 67](#_Toc151975590)

[12.5.1 t451\_report\_connect(chassis\_idx, board\_idx) 67](#_Toc151975591)

[12.5.2 t451\_report\_disconnect(chassis\_idx, board\_idx) 68](#_Toc151975592)

[12.5.3 t451\_report\_overload(chassis\_idx, board\_idx) 68](#_Toc151975593)

[12.5.4 t451\_report\_underload(chassis\_idx, board\_idx) 68](#_Toc151975594)

[12.5.5 t451\_report\_shortcircuit(chassis\_idx, board\_idx) 69](#_Toc151975595)

[12.5.6 t451\_report\_loading(chassis\_idx, board\_idx) 69](#_Toc151975596)

[13 Callback函數 70](#_Toc151975597)

[14 示例 71](#_Toc151975598)

[13.1 获取板卡，端口信息 71](#_Toc151975599)

[13.2 收送包，获取端口Stream数据 73](#_Toc151975600)

[13.3 撷取包，并呈现包信息 74](#_Toc151975601)

[13.4 以UI方式呈现收送包 78](#_Toc151975602)

[13.5 T451执行connect测试 78](#_Toc151975603)

[15 缺漏(未来版本补齐) 83](#_Toc151975604)

# 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*

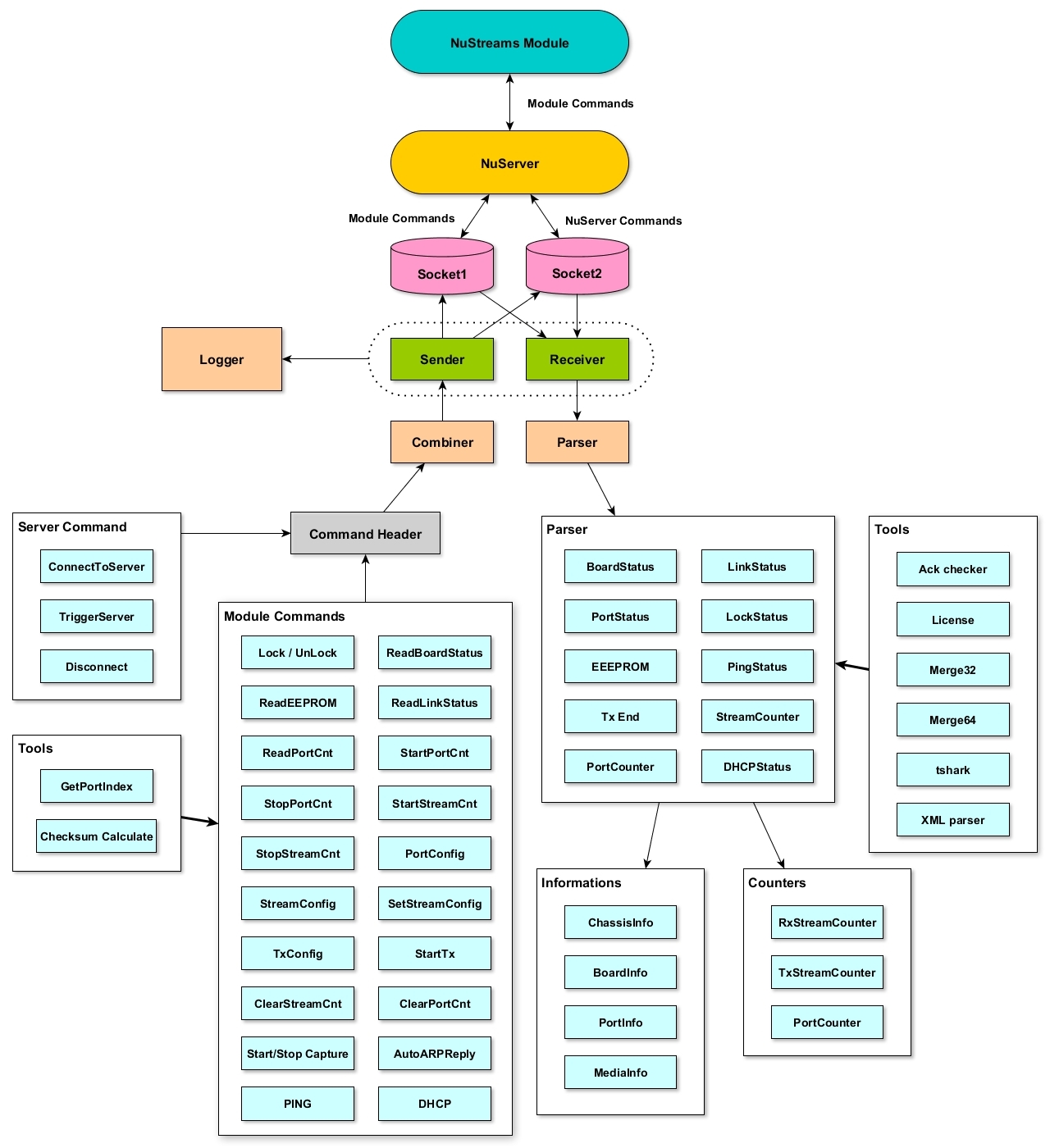
本文章接下来介绍NuPython的架构，并提供编程建议。

# 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

1. **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

1. **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

1. **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()*

1. **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

1. **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()*

1. **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)*

1. **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)*

1. **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)*

1. **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)*

1. **端口解锁**

*nscmd.port\_unlock()*

1. **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。

# 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。

## 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)* |

## server\_disconnect()

|  |  |
| --- | --- |
| Function | 断开和NuServer连线 |
| Input | None |
| Output | None |
| Note |  |
| Example | *self.server\_disconnect()* |

## 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)* |

## 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()* |

## 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)* |

# 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。

## 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))* |

## 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))* |

## 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))* |

## 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)* |

## 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)* |

## 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)* |

## 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)* |

## 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)* |

# 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)* |

# 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()* |

# 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)* |

# 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()* |

# 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)* |

# 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)* |

# 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數量 |
| 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內容，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)* |

# 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)* |

# 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: 函數名，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：

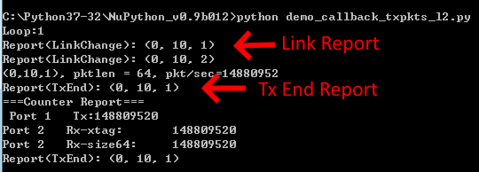
1. 客户端宣告回调函数: 用戶自行處理函數內部，範例只是顯示

|  |
| --- |
| 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. 将回调函数传入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。



# 示例

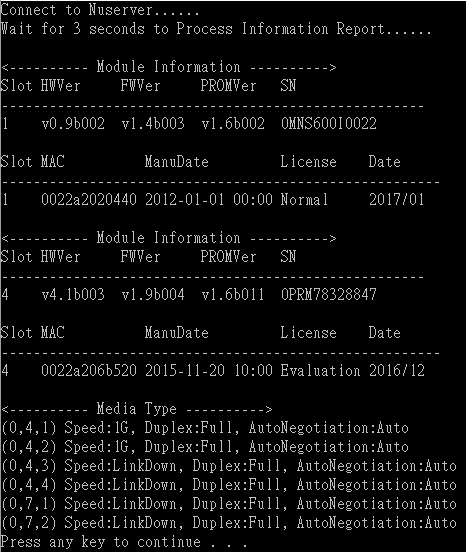
本章节展示几个示例，更多示例在发行的NuPython底下demo档案夹。

## 14.1 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：

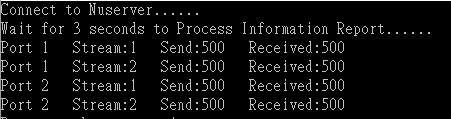


## 14.2 收送包，获取端口Stream数据

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：



## 14.3 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()* |

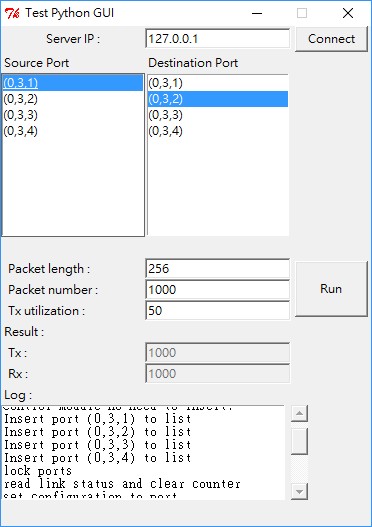
結果:



## 14.4 以UI方式呈现收送包

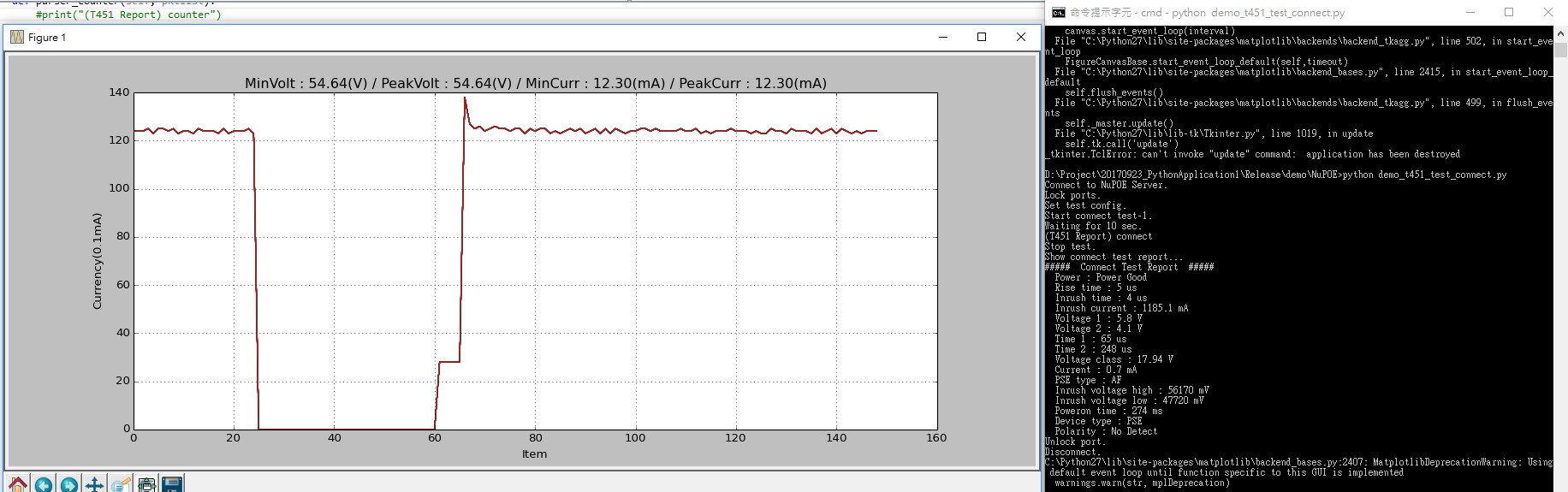
利用Python內建的套件 - Tkinter(Python2.7)。來呈現UI。

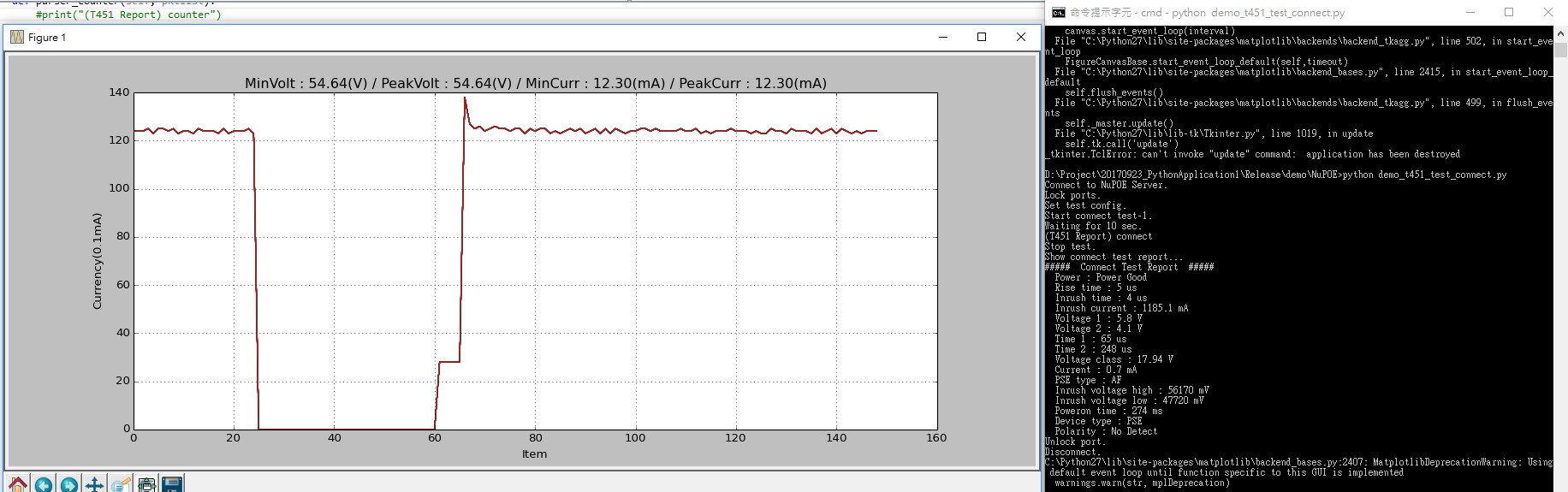
結果:



## 14.5 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。





|  |
| --- |
| 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.counter\_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\_idx]  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() |

# 缺漏(未来版本补齐)

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