คู่มือคำสั่ง

Smart Box DAQ Command

(SD-Command)

รุ่น V17 (FW 0.74.3)

บริษัท อาร์ แอนด์ ดี คอมพิวเตอร์ ซิสเท็ม จำกัด

SD-Command_Manual_V17_R241204

สารบัญ

แนะนำการใช้งาน

นอกจากการใช้งาน Octopus Smart Box บน Node-RED ผ่าน Octopus Smart Box Node แล้ว ผู้ใช้งานยังสามารถเขียนโปรแกรมด้วยภาษาคอมพิวเตอร์อื่น เพื่อสื่อสารกับ Octopus Smart Box ได้โดยตรง โดยใช้ชุดคำสั่ง Smart Box DAQ Command หรือ (SD-Command) ซึ่งจะอธิบายรายละเอียดเพิ่มเติมในส่วนถัดไป

ชุดคำสั่ง SD-Command นี้ สามารถใช้งานได้กับ Octopus Smart Box รุ่น DAQ-01 Series ทั้ง DAQ-01, DAQ-01A และ DAQ-01S โดยผ่านทาง USB หรือ Ethernet ยกเว้นสำหรับ DAQ-01 ที่ไม่มี Ethernet จึงสั่งได้เฉพาะทาง USB

การสื่อสารผ่าน USB กับ Octopus Smart Box จะทำงานในโหมด USB CDC (virtual COM port) ซึ่งปรากฏใน Windows เป็น Serial Port ที่แสดงเป็น COMxx บน Windows Device Manager อย่างไรก็ตาม การสื่อสารผ่าน virtual COM port นั้นต่างจาก Serial Port ปกติตรงที่ไม่จำเป็นต้องกำหนดค่า baudrate, parity หรือ stop bit เนื่องจาก virtual COM port ไม่ได้ใช้ค่าดังกล่าว และสามารถส่งข้อมูลได้เร็วกว่า Serial Port แบบปกติเนื่องจากใช้การส่งผ่าน USB โดยตรง

การสื่อสารผ่าน Ethernet กับ Octopus Smart Box ใช้ TCP/IP ในลักษณะเดียวกับ Telnet โดยคำสั่งที่ใช้จะเหมือนกับการสื่อสารผ่าน USB แต่ต้องทำการ connect socket ก่อน โดยระบุ IP และ port ของอุปกรณ์ที่ตั้งค่าไว้ หลังจากใช้งานเสร็จสิ้นต้องทำการ disconnect และการส่งข้อมูลจะใช้การส่งผ่าน TCP/IP Socket

คำสั่งในการติดต่อสื่อสาร

คำสั่งในการสื่อสารกับ Octopus Smart Box จะอยู่ในรูปแบบตัวอักษร ASCII และการตอบกลับส่วนใหญ่จะเป็น ASCII ด้วย ยกเว้นในบางกรณีที่ต้องการสแกนอ่านข้อมูลบางฟอร์แมต จะส่งข้อมูลกลับมาเป็น Binary เพื่อเพิ่มความเร็วในการสื่อสาร

รูปแบบของคำสั่ง

:Command [arg0] [arg1]<cr>

มีลักษณะเป็น Command หรือคำสั่ง ตามหลังอักษร ':' และอาจมี argument ตามกี่ตัวขึ้นอยู่กับคำสั่ง โดยคั่นระหว่าง Command และ argument แต่ละตัวด้วยช่องว่าง และปิดท้ายคำสั่งด้วย <cr> คือ carriage return หรือรหัส ASCII 13 เพื่อแสดงว่าจบคำสั่ง

การตอบสนองคำสั่ง

ทุกคำสั่งที่สั่ง หากทำงานได้จะมีการตอบสนองโดยส่งคำสั่งเดิมคืนมา และหากมีข้อมูลตามมาด้วย เช่น

สั่ง :info 1

ตอบ :info 1 SmartBox

หากคำสั่งที่ส่งไปไม่ถูกต้องหรือไม่สามารถทำงานได้ จะไม่มีการตอบกลับใดๆ สำหรับการแสดงตัวอย่างคำสั่ง จะใช้อักษร '>' นำหน้าเพื่อแสดงคำสั่งที่ส่งออกไป และใช้อักษร '<' นำหน้าเพื่อตอบสนองคำสั่งที่ได้รับ

สรุปรายการคำสั่ง

คำสั่ง	คำอธิบาย
คำสั่งแสดงข้อมูลอุปกรณ์	
:info arg0	แสดงข้อมูลเกี่ยวกับอุปกรณ์ โดยมี arg0 เพื่อกำหนดข้อมูลที่ต้องการ
:serialnum	แสดง serial number ของอุปกรณ์
:devname	อ่านหรือตั้งชื่อให้กับอุปกรณ์
คำสั่ง Digital IO	
:endo	กำหนดพอร์ต Digital ทั้ง 8 ช่อง ให้ช่องไหนเป็น output
:endob	กำหนดพอร์ต Digital ช่องไหนเป็น output โดยกำหนดทีละช่อง
:doutbeglow	กำหนดพอร์ต Digital ทั้ง 8 ช่อง ให้ช่องไหนเป็น output และเป็น low ตอนเปิดเครื่อง
:din	อ่านข้อมูล Digital ทั้ง 8 ช่อง ไม่ว่าช่องนั้นเป็น input หรือ output ก็ตาม
:dinb	อ่านข้อมูล Digital จากช่องที่ระบุ
:dout	สั่งให้พอร์ต Digital ทั้ง 8 ช่อง เฉพาะช่องที่ตั้งเป็น output ไว้เป็น low หรือ high
:doutb	สั่งให้ digital IO ช่องไหนเป็น low หรือ high โดยช่อง IO นั้นต้องตั้งเป็น output ด้วย
:pfimode	กำหนดโหมดในการทำงานให้กับ pfi 0-3 ซึ่งตรงกับช่อง Digital 0-3
:pfi	อ่านค่าจากช่อง pfi ใช้สำหรับอ่านค่า counter และ rate
:pwm	กำหนดค่า pwm ให้กับช่อง pwm 0-1
:pwmrate	กำหนดความถี่ pwm ไปเป็นค่าอื่นนอกจากค่าเริ่มต้น
คำสั่ง Analog In	
:ain	อ่านพอร์ต Analog ช่องที่ต้องการ
คำสั่งตั้งค่าการติดต่อสื่อสาร Ethernet	
:ipaddr	ตั้งค่า IP address ให้อุปกรณ์
:netmask	ตั้งค่า sub netmask
:gateway	ตั้งค่า gateway IP address
:dns	ตั้งค่า dns server
:port	ตั้งค่า port ในการติดต่อสื่อสารแบบ Ethernet เพื่อส่งคำสั่ง
:mac	อ่านค่า mac address ของอุปกรณ์
:dhcp	กำหนดให้ทำงานในโหมด dhcp หรือไม่
การสแกนอ่านข้อมูล	
:encode	กำหนดรูปแบบการส่งข้อมูลเมื่อสั่งสแกนอ่าน
:slist	เพิ่มรายการข้อมูลที่ต้องการสแกนอ่าน โดยเพิ่มทีละช่องเป็นลำดับ
:slists	กำหนดรายการข้อมูลที่ต้องการสแกนอ่านในคราวเดียวกัน
:dec	กำหนดค่า over sampling
:srate	กำหนดอัตราในการสแกน
:scan	สั่งให้เริ่มต้นสแกนอ่านข้อมูลตามที่ตั้งค่าไว้
:stop	สั่งให้หยุดสแกนอ่าน

คำสั่งใช้งาน Modbus	
:serial_mode	กำหนดโหมดการสื่อสารให้กับ Modbus RS485
:serial_baud	กำหนดความเร็วในการสื่อสารให้กับ Modbus RS485
:mb485_mode	กำหนด Modbus RS485 ให้ทำงานในโหมดไหน
:mb_addr	กำหนด Modbus Slave ID ระหว่าง 1-247
:mbtcp_mode	กำหนด Modbus TCP ให้ทำงานหรือไม่
:mbtcp_port	กำหนดหมายเลข port สำหรับ Modbus TCP
คำสั่งเกี่ยวกับเวลา	
:ymd	กำหนดปี,เดือน,วัน ให้กับอุปกรณ์
:hms	กำหนดเวลาชั่วโมง,นาที,วินาที ให้กับอุปกรณ์
คำสั่งอื่นๆ	
:getsw1	อ่านค่าจาก SW1 ว่ามีการกดหรือไม่
:showled	แสดง RGB Led ให้เป็นสีที่ต้องการ เป็นระยะเวลาตามที่กำหนด
:setcounter	กำหนดค่า counter เริ่มต้น
:reset	ทำการ reset ค่าในอุปกรณ์ เช่น counter ให้เป็น 0
คำสั่งใช้งาน MQTT	
:mqtt_enable	กำหนดว่าให้ใช้งาน mqtt หรือไม่
:mqtt_server	กำหนด url หรือ ip ของ mqtt server
:mqtt_id	กำหนด mqtt device id
:mqtt_user	กำหนด mqtt user
:mqtt_pswd	กำหนด mqtt password
:mqtt_topic	กำหนด topic ของ mqtt
:mqtt_data_key	กำหนดชื่อ key สำหรับ data
:mqtt_ts_key	กำหนดชื่อ key สำหรับ timestamp
:mqtt_port	กำหนดค่า port ในการติดต่อสื่อสาร mqtt ให้เป็นค่าอื่น (จากค่าเริ่มต้นคือ 1883)
:mqtt_qos	กำหนด QoS ในการสื่อสาร
:mqtt_interval	กำหนดอัตราในการส่งหน่วยเป็นวินาที
:mqtt_keepalive	กำหนดเวลาในการส่ง keepalive หน่วยเป็นวินาที
:alists	กำหนดรายการข้อมูล Analog ที่ต้องการส่งผ่าน internet
:dilsts	กำหนดรายการข้อมูล Digital ที่ต้องการส่งผ่าน internet

คำอธิบายคำสั่ง

คำสั่งแสดงข้อมูลอุปกรณ์

info

แสดงข้อมูลเกี่ยวกับอุปกรณ์ (information)

Command :info <type>

; <type> คือตัวเลขชนิดข้อมูล มีดังนี้

; 1 = แสดงชื่อผลิตภัณฑ์ (product name)

; 2 = แสดงโมเดลของผลิตภัณฑ์ (product model)

; 3 = แสดงรุ่น (version) ของ firmware เป็นตัวเลข 6 หลัก คือ

; <mmnnss> โดย mm=major, nn=minor, ss=subver

Response :info <type> <str>

; <str> คือข้อมูล

Example

> :info 1

< :info 1 SmartBox ;ชื่อผลิตภัณฑ์คือ SmartBox

> :info 2

< :info 2 DAQ-01A ;โมเดลคือ DAQ-01A

> :info 3

< :info 3 012345 ; แสดง firmware รุ่น 1.23.45

serialnum

แสดงข้อมูล serial number

Command :serialnum

Response :serialnum <S/N>

; <S/N> คือหมายเลข serial number ของอุปกรณ์

Example

> :serialnum

< :serialnum 082001024 ; serial number= 082001024

devname

ตั้งชื่อ หรือแสดงชื่อ device name (เป็นชื่อที่ใช้เรียก device และใช้ในการเชื่อมต่อ)

Command :devname <name>

; <name> คือชื่อ device name ที่ต้องการตั้ง หากไม่ได้ใส่ จะเป็นการนำชื่อเดิมที่ตั้งไว้มาแสดง ชื่อ device name จะต้องไม่มีช่องว่าง และไม่มีตัวอักษร # : , และความยาวรวมกันไม่เกิน 30 ตัวอักษร

Response :devname <name>

; <name> คือชื่อเดิม หรือชื่อที่ตั้งไว้

Example

> :devname my_daq_device

< :devname my_daq_device

คำสั่ง Digital IO

endo

กำหนดพอร์ต Digital IO ทั้ง 8 ช่อง ให้ช่องไหนเป็น Output (Enable Digital Out)

Command : endo <val>

; <val> คือค่าในการกำหนดพอร์ตแต่ละบิตสำหรับพอร์ตช่องไหนให้เป็น output

; บิตใดเป็น 1 คือพอร์ตช่องนั้นเป็น output

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :endo 17 ; 17=0x11=00010001b กำหนดให้พอร์ตช่อง D0,D4 เป็น output ส่วนช่องที่เหลือคือ D1,D2,D3,D5,D6,D7 เป็น input

< :endo 17

endob

กำหนดพอร์ต Digital IO ให้ช่องไหนเป็น Output โดยกำหนดทีละช่อง (Enable Digital Out Bit)

Command : endob <chn> <val>

; <chn> มีค่า 0-7 คือช่องพอร์ตที่ต้องการกำหนด

; <val> คือค่าในการกำหนด 0=input, 1=output

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :endob 4 1 ; กำหนดให้พอร์ตช่อง D4 เป็น output

< :endob 4 1

> :endob 0 0 ; กำหนดให้พอร์ตช่อง D0 เป็น input

< :endob 0 0

doutbeglow

กำหนดพอร์ต Digital IO ทั้ง 8 ช่อง ให้ช่องไหนเป็น Output และเป็น Low ตอนเปิดเครื่อง (Digital Output Begin Low)

Command :doutbeglow <val>

; <val> คือค่าในการกำหนดพอร์ตแต่ละบิตสำหรับพอร์ตช่องใดให้เป็น output

; บิตใดเป็น 0 คือช่องนั้นเป็น Output และเป็น low ตอนเปิดเครื่อง

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :doutbeglow 240 ; 240=0xF0=11110000b กำหนดให้พอร์ตช่อง D4,D5,D6,D7 เป็น Output Low ตอนเปิดเครื่อง ส่วนช่องที่เหลือ D0,D1,D2,D3 เป็น Input

< :doutbeglow 240

➢ ***ค่า <val> ของคำสั่ง doutbeglow แต่ละบิตจะมีความหมายตรงข้ามกับคำสั่ง endo กล่าวคือ ค่าบิตเป็น 0 จะเป็น output แต่คำสั่ง endo จะเป็น input นอกจากนี้ คำสั่ง doutbeglow สามารถจำค่าที่ตั้งไว้ได้แม้ในกรณีที่ไฟดับหรือรีสตาร์ทระบบ แต่คำสั่ง endo จะไม่จำค่าดังกล่าวเมื่อเกิดไฟดับหรือรีสตาร์ท***

din

อ่านค่าจากพอร์ต Digital IO ทุกช่อง (Digital Input)

Command :din

Response :din <val>

; <val> มีค่า 0-255 เมื่อแปลงเป็น binary 8 bit จะเป็นข้อมูลที่อ่านได้จากพอร์ต Digital แต่ละช่อง

Example

> :din

< :din 52 ; 52=0x34=00110100b พอร์ตช่อง D2,D4,D5 อ่านได้เป็น High ส่วนช่องที่เหลือ D0,D1,D3,D6,D7 อ่านได้เป็น Low

dinb

อ่านค่าจากพอร์ต Digital จากช่องที่ระบุ (Digital Input Bit)

Command :dinb <chn>

; <chn> มีค่า 0-7 คือช่องพอร์ตที่ต้องการอ่าน

Response :dinb <chn> <val>

; <val> คือค่าที่อ่านได้ เป็น 0=Low, 1=High

Example

> :dinb 5 ; อ่านช่อง D5

< :dinb 5 0 ; ช่อง D5 มีค่าเป็น 0=Low

dout

สั่งให้พอร์ต Digital ทั้ง 8 ช่อง (เฉพาะที่ตั้งเป็น output ไว้) เป็น Low หรือ High (Digital Out)

Command :dout <val>

; <val> คือค่าในการกำหนดพอร์ตแต่ละบิตสำหรับแต่ละช่อง เป็น Low=0 หรือ High=1

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :dout 63 ; 63=0x3F=00111111b กำหนดให้พอร์ตช่อง D6,D7 เป็น Low (หาก enable ไว้) ส่วนช่องที่เหลือ D0,D1,D2,D3,D4,D5 เป็น High

< :dout 63

doutb

สั่งให้พอร์ต Digital ช่องที่กำหนด เป็นค่าตามต้องการ (Digital Out Bit)

Command :doutb <chn> <val>

; <chn> มีค่า 0-7 คือช่องที่ต้องการสั่ง, <val> ค่าที่ต้องการ 0=Low 1=High

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :doutb 6 0 ; ให้พอร์ตช่อง D6 เป็น Low

< :doutb 6 0

pfimode

กำหนดโหมดในการทำงานให้กับช่อง pfi 0-3 ซึ่งตรงกับช่อง Digital 0-3

Command :pfimode <chn> <mode>

; <chn> มีค่า 0-3 คือช่อง pfi0-3 (D0-3)

; <mode> คือโหมดการทำงาน มีดังนี้

; 0=digital in (pfi0-3)

; 1=counter (เฉพาะ pfi0-1)

; 4=rate (เฉพาะ pfi2-3)

; 2,3 (สำรองไว้สำหรับ edge trigger)

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :pfimode 1 1 ; ให้ pfi 1 หรือ D1 เป็น counter

< :pfimode 1 1

> :pfimode 2 4 ; ให้ pfi 2 หรือ D2 เป็น rate

< :pfimode 2 4

pfi

อ่านค่าจากพอร์ต pfi

Command :pfi <chn>

; <chn> มีค่า 0-3 คือช่อง pfi

Response :pfi <chn> <val>

; <val> มีค่าที่อ่านได้ เป็นไปตามโหมดที่ตั้งไว้

Example

> :pfimode 2 4 ; ให้ pfi2 เป็น rate

< :pfimode 2 4 ; echo

> :pfi 2 ; อ่าน

< :pfi 2 2345 ; อ่าน pfi 2 ได้ค่า rate=2345

pwm

กำหนดค่า pwm ให้กับช่อง pwm 0-1

ซึ่งตรงกับพอร์ต D6,7 ตามลำดับ, การสั่ง pwm นี้ เราไม่จำเป็นต้องตั้งโหมด สั่งออกได้เลย

Command :pwm <chn> <val>

; <chn> มีค่า 0-1 คือช่อง pwm

; <val> ค่า duty cycle ของ pwm มีค่า 0-1023, โดยค่า 1023 คือค่าสูงสุด, 0 คือค่าต่ำสุด

; หากสั่งเป็น 0 จะเป็นการยกเลิกการใช้งาน pwm, เท่ากับสั่ง Digital Out เป็น 0

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :pwm 1 600 ; ให้ pwm1 (D7) มีค่า duty cycle = 600/1023

pwmrate

กำหนดความถี่ pwm (PWM Rate), ปกติค่าความถี่ pwm คือ 1.46 kHz สามารถตั้งให้เป็นความถี่อื่นได้

Command :pwmrate <chn> <val>

; <chn> มีค่า 0-1 คือช่อง pwm

; <val> คือระดับความถี่ มีค่า 1-4 ดังนี้

; 1 = 366 Hz

; 2 = 1.46 kHz (ค่าเริ่มต้น)

; 3 = 5.86 kHz

; 4 = 23.4 kHz

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :pwmrate 1 3 ; ให้ pwm1 (D7) ส่งออกมาด้วยความถี่ 5.86 khz

< :pwmrate 1 3

คำสั่ง Analog In

ain

อ่านค่าพอร์ต Analog In จากช่องที่ต้องการ (Analog In)

Command :ain <chn>

; <chn> มีค่า 0-7 คือช่องพอร์ตที่ต้องการอ่าน

Response :ain <chn> <val>

; <val> คือค่าที่อ่านได้

Example

> :ain 3 ; อ่านค่าพอร์ต A3

< :ain 3 2.345 ; ช่อง A3 มีค่า 2.345V

คำสั่งตั้งค่าการติดต่อสื่อสาร Ethernet

ipaddr

ตั้งค่า IP address ให้กับอุปกรณ์

Command :ipaddr <n.n.n.n>

; <n.n.n.n> คือค่า IP address ที่ต้องการ

; หากไม่ได้ใส่ จะเป็นการอ่านค่า IP address ปัจจุบันออกมา

Response :ipaddr <n.n.n.n>

; <n.n.n.n> คือค่า IP address ล่าสุด

Example

> :ipaddr ; อ่าน IP address

< :ipaddr 192.168.1.123 ; ตอบค่า IP ล่าสุด

> :ipaddr 192.168.1.111 ; ตั้งค่า IP address เป็น 192.168.1.111

< :ipaddr 192.168.1.111 ; ตอบค่า IP ล่าสุด

netmask

ตั้งค่า sub netmask ให้กับอุปกรณ์, คำสั่ง netmask ใช้เหมือนคำสั่ง ipaddr

gateway

ตั้งค่า internet gateway ให้กับอุปกรณ์, คำสั่ง gateway ใช้เหมือนคำสั่ง ipaddr

dns

ตั้งค่า dns ให้กับอุปกรณ์, คำสั่ง dns ใช้เหมือนคำสั่ง ipaddr

port

ตั้งค่า port ให้กับอุปกรณ์ (เป็น command port ที่ใช้ส่งคำสั่ง)

Command :port <val>

; <val> คือหมายเลข port ที่ต้องการ

; หากไม่ได้ใส่ จะเป็นการอ่านค่า port ปัจจุบันออกมา

Response :port <val>

; <val> คือค่า port ล่าสุด

Example

> :port ; อ่านค่า port ที่ตั้งไว้

< :port 5555 ; ค่าที่อ่านได้

mac

อ่านค่า mac address ของอุปกรณ์

Command :mac

Response :mac <mac_addr>

; <mac_addr> คือค่า mac address ของอุปกรณ์

Example

> :mac ; อ่านค่า mac address

< :mac 72:64:71:7C:4D:6F ; ค่าที่อ่านได้

dhcp

กำหนดให้ทำงานในโหมด dhcp หรือไม่

Command :dhcp <val>

; <val> มีค่า 0 หรือ 1, หากไม่ได้ระบุ จะเป็นการอ่านค่าที่ตั้งไว้ออกมา

; 0=off คือไม่ใช้ dhcp, ใช้แบบ fixed IP ตามค่า IP ที่ตั้งไว้

; 1=on คือใช้ dhcp, จะมีการร้องขอ IP กับ dhcp server ในวงแลน

Response :dhcp <val>

; <val> คือค่า dhcp ล่าสุดที่ตั้งไว้

Example

> :dhcp ; อ่านค่า dhcp

< :dhcp 0 ; 0 = ไม่ได้ใช้แบบ dhcp (เป็น fixed IP)

> :dhcp 1 ; ตั้งให้ใช้ dhcp

< :dhcp 1 ; ค่า dhcp ล่าสุด

➢ ***เมื่อตั้งค่า dhcp เป็น 1 แล้วรอสักครู่ จะได้ค่า IP ค่าใหม่จาก dhcp server โดยสามารถดู IP ที่ได้รับด้วยคำสั่ง ipaddr ตามที่กล่าวมา***

การสแกนอ่านข้อมูล

การสแกนอ่านข้อมูล คือการสั่งให้ Smart Box ทำการส่งข้อมูลออกมาโดยอัตโนมัติตลอดเวลาในลักษณะของการสแกนอ่าน โดยเราจะต้องระบุ ข้อมูลที่ต้องการ, รูปแบบการส่งข้อมูล, อัตราการส่ง แล้วจึงสั่งเริ่มต้นสแกน ทำโดยใช้คำสั่งต่อไปนี้

encode

กำหนดรูปแบบการส่งข้อมูลเมื่อสั่งสแกนอ่าน

Command :encode <type>

; <type> คือวิธีการ encode มีดังนี้

; 0 = binary ไม่มี header

; 1 = ASCII คั่นด้วย ‘,’

; 4 = binary มี header

; 129 = เหมือน encode=1 แต่มี Timestamp แบบ double precision สำหรับ Excel

; 33 = เหมือน encode=1 แต่มี Timestamp แบบ ISO 8601

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :encode 1 ; กำหนด encode=1

< :encode 1

slist

เพิ่มรายการข้อมูลที่ต้องการสแกนอ่าน โดยเพิ่มทีละช่องเป็นลำดับ

Command :slist <index> <chn>

; <index> คือลำดับข้อมูลที่ต้องการสแกน เริ่มจาก 0 ไปทีละหนึ่ง

; เมื่อ <index>=0 จะเริ่มต้นนับจำนวนรายการใหม่เป็น 1 ช่อง

; จากนั้นคำสั่ง slist ถัดไป เราต้องเพิ่ม <index> ทีละ 1

; <chn> คือชนิดข้อมูลที่ต้องการสแกน มีดังนี้

; 0-7 คือ Analog 0-7

; 8 คือ Digital In

; 9, 10 คือ Counter 0, 1 ตามลำดับ

; 11,12 คือ Rate 0, 1 ตามลำดับ

; *** ชนิดข้อมูลในรายการ จะต้องเรียงจากน้อยไปหามาก เช่นใส่ชนิดที่ 1 หลังชนิดที่ 2 ไม่ได้

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :slist 0 0 ; เริ่มรายการแรกคือ A0

< :slist 0 0

> :slist 1 8 ; รายการที่สองคือ Digital In ทั้ง 8 ช่อง

< :slist 1 8

> :slist 2 11 ; รายการที่สามคือ Rate 0

< :slist 2 11

slists

กำหนดรายการข้อมูลที่ต้องการสแกนอ่าน ในคราวเดียวกัน

Command :slists <multi_chn>

; <multi_chn> คือรายการข้อมูลทั้งหมดที่ต้องการสแกน โดยใช้ 1 บิตต่อช่องในลำดับเดียวกัน เช่น บิต 0-7 คือ <chn> 0-7 ของ slist ซึ่งคือ Analog, บิต 9 คือ ช่อง 9 ของ slist คือ Digital เป็นต้น โดยการสแกนข้อมูล ก็จะเริ่มจากช่องลำดับน้อยไปหามาก

Response :slists <multi_chn>

Example

> :slist 0 0	; ใช้ slist เพิ่มรายการแรกคือ A0
< :slist 0 0	; command echo
> :slists	; ถาม slists
< :slists 1	; 1=bit 0 คือ chn=0 คือ A0
> :slists 513	; 513=0x201=0 0010 0000 0001 คือรายการสแกนได้แก่ A0 กับ counter 0
< :slists 513	; command echo

dec

กำหนดค่า over sampling (decimal)

การ over sampling คือการอ่านข้อมูลหลายค่าก่อน แล้วนำค่ามาเฉลี่ยเพื่อให้ได้ข้อมูลที่มีความละเอียดแม่นยำขึ้น และกรองสัญญาณรบกวนความถี่สูงในข้อมูลได้ แต่จะทำให้ความเร็วในการอ่านข้อมูลช้าลง

Command :dec <val>

; <val> มีค่า 1-1024 คือค่า over sampling, หากไม่ระบุ จะแสดงค่าล่าสุด

Response :dec <val>

Example

> :dec 256	; กำหนดค่า decimal เป็น 256
< :dec 256	; command echo

srate

กำหนดอัตราในการสแกน (scan rate)

อัตราการในการสแกน คือความถี่ในการอ่านสัญญาณ Analog มีหน่วยเป็น 0.1 millisecond (ms) แต่อัตราในการส่งข้อมูล จะสัมพันธ์กับค่า decimal หรือ over sampling ด้วย คือ

อัตราในการส่งข้อมูล = อัตราในการสแกน หารด้วยค่า decimal

ตัวอย่างเช่น srate = 10 (1kHz), dec = 100

อัตราในการส่งข้อมูล = 1000 / 100

= 0.1 sec หรือ 10 ข้อมูลต่อวินาที

Command :srate <val>

; <val> มีค่า 2-65535 คือความเร็วในการอ่าน หน่วยเป็น 0.1 millisecond (ms) หรือ 0.0001 วินาที เช่น 10 คือ 1 ms, 10000 คือ 1 วินาที (sec)

Response :srate <val>

Example

> :srate 10 ; กำหนดค่า scan rate=10 คือ 10x0.1 = 1ms (1kHz)

> :srate 10 ; command echo

scan

สั่งให้เริ่มต้นการสแกนอ่านข้อมูล

หลังจากที่ได้ระบุข้อมูลที่ต้องการ รูปแบบการสแกน และอัตราการสแกน เมื่อสั่ง scan ก็จะเริ่มทำการอ่านและส่งข้อมูลออกมาโดยอัตโนมัติ

Command :scan <type>

; <type> มีค่าดังนี้

; 0 คือ สแกนแบบปกติ

; 9 คือ สแกนแบบอัตโนมัติ เมื่อไฟดับแล้วติด ก็กลับมาสแกนต่อ

; คำสั่ง scan 9 ใช้สำหรับกรณีที่ทำงานอิสระ เช่นส่งข้อมูลไป server โดยใช้ http

Response ไม่มี, คำสั่งนี้เมื่อสั่งแล้วจะไม่ echo คำสั่ง แต่จะเริ่มทำการสแกนแล้วส่งข้อมูลออกมา

Example

> :scan 0 ; เริ่มต้นสแกน หลังจากที่ตั้ง slist ไว้แล้ว

< เริ่มส่งข้อมูลที่สแกนได้ออกมา

stop

สั่งให้หยุดการสแกน

Command :stop

Response :stop

Example

> :stop ; หยุดการสแกน

< :stop ; command echo

รูปแบบการส่งข้อมูลจากการสแกน

เมื่อสั่งสแกนอ่านข้อมูล จะถูกส่งมาต่อเนื่องโดยอัตโนมัติ โดยส่งเฉพาะข้อมูลที่ระบุใน slist หรือ slists โดยรูปแบบข้อมูลขึ้นอยู่กับ encode ที่ตั้งไว้

กรณีส่งแบบ ASCII

Encode 1 คือการส่งแบบ ASCII แต่ละช่องคั่นด้วย ‘,’ และปิดท้ายข้อมูลด้วย <cr> หรือ ASCII 13

โดย A0-A7 จะส่งค่าแรงดันเป็น Volt ซึ่งเป็นเลขทศนิยม

ค่า Digital ส่งเป็นตัวเลข 0-255 เหมือนคำสั่ง **‘din’**,

ค่า Counter และ Rate ส่งเป็นตัวเลขไม่มีทศนิยม

Encode 129 ส่งแบบ Encode 1 แต่เพิ่ม Timestamp นำหน้าข้อมูล เป็นเลขทศนิยมแบบ double precision ที่ Excel ใช้อ่านได้

Encode 33 ส่งแบบ Encode 1 แต่เพิ่ม Timestamp นำหน้าข้อมูล เป็นแบบ ISO 8601 ตัวอย่างเช่น 2024-01-23T15:30:21.497, แล้วตามด้วยข้อมูล

กรณีส่งแบบ Binary

การส่งแบบ binary คือการส่งในรูปข้อมูล binary ต่อเนื่องกันไปตามที่ระบุใน slist ส่งแบบ little endian คือส่ง low byte ก่อน

ช่อง Analog A0-A7 จะส่งเป็นข้อมูล 2 byte เป็นเลข 16bit Integer ที่มีค่าระหว่าง -32768 ถึง 32767 โดยค่า -32768 คือค่าต่ำสุด -10.0V และค่า 32,767 คือค่า 9.9997V (เพราะ 10V คือ 32768)

การแปลงเป็นค่าแรงดัน จะใช้สูตร

Vout = (val * 10.0) / 32768 ; โดยที่ val คือค่าที่อ่านได้

ช่อง Digital ส่งข้อมูล 2 byte แต่ใช้เฉพาะ byte แรก สำหรับ D0-D7

ช่อง Counter ส่งข้อมูล 4 byte เป็นค่า counter ขนาด 32 bit

ช่อง Rate ส่งเป็นข้อมูล 2 byte ขนาด 16 bit

Encode 0 จะถูกส่งในรูปแบบ binary โดยไม่มี Header ข้อมูลในแต่ละช่องจะถูกส่งต่อเนื่องกันไป ซึ่งเป็นวิธีที่ง่ายที่สุดในการส่งข้อมูล แต่มีข้อเสียคือ การแยกข้อมูลแต่ละชุดออกจากกันทำได้ยาก

Encode 4 ส่งแบบ binary มี Header เป็น packet มีลักษณะดังนี้

AA CC <len_h> <len_l> <chn> <chn> <chn> ...

<len_h>,<len_l> คือความยาวของข้อมูลที่ตามมาขนาด 2 ไบต์ จะส่ง high byte ก่อน (ซึ่งมักเป็น 0 เนื่องจากขณะนี้ความยาวยังไม่เกิน 255 ไบต์) ตามด้วย low byte

<chn> คือข้อมูลแต่ละช่องที่ตามมา มีฟอร์แมตดังนี้

En bl bh ; En ตามด้วยข้อมูล 2 byte ไบต์ low และไบต์ high โดยที่

; n=0-7 คือช่อง Analog 0-7

; n=8 คือช่อง Digital

; n=9,10 คือช่อง Rate D2 และ D3 ตามลำดับ (ค่านี้จะไม่เหมือนกับ slist)

Dn bl bh bu bm ; Dn ตามด้วยข้อมูล 4 byte จาก low ไป high

; n=0, 1 คือ Counter D0 และ D1 ตามลำดับ

คำสั่งใช้งาน Modbus

เป็นกลุ่มคำสั่งที่ช่วยในการติดตั้งค่าในการใช้งาน Modbus RS485 (Master/Slave) และ TCP (Slave) สำหรับ Modbus ที่ทำงานอยู่บน RS485 ซึ่งส่งแบบ Serial จึงมีคำสั่งในการตั้งโหมดและ baudrate ของ Serial ด้วย

serial_mode

กำหนดโหมดการทำงานของ Serial สำหรับ Modbus RS485

Command :serial_mode <mode>
; <mode> คือโหมดของ serial มีดังนี้
; 0 คือ N,8,1 (No parity, 8 data bits, 1 stop bit)
; 1 คือ E,8,1 (Even parity, 8 data bits, 1 stop bit)
; 2 คือ O,8,1 (Odd parity, 8 data bits, 1 stop bit)
; 3 คือ N,8,2 (No parity, 8 data bits, 2 stop bit)
; 4 คือ E,8,2 (Even parity, 8 data bits, 2 stop bit)
; 5 คือ O,8,2 (Odd parity, 8 data bits, 2 stop bit)

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :serial_mode 0 ; สั่ง serial ทำงานในโหมด 0
< :serial_mode 0 ; command echo

serial_baud

กำหนด baudrate ให้กับ Serial สำหรับ Modbus RS485

Command :serial_baud <baud>
; <baud> คือ baudrate มีค่าตั้งแต่ 300-460800

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :serial_mode 0 ; สั่ง serial ทำงานในโหมด 0
< :serial_mode 0 ; command echo

mb485_mode

กำหนดโหมดการทำงานของ Modbus บน RS485

Command :mb485_mode <mode>
; <mode> คือโหมดการทำงานมีดังนี้
; 0 = Disable, ปิดการทำงาน Modbus RS485
; 1 = Master
; 2 = Slave RTU
; 3 = Slave ASCII

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :mb485_mode 2 ; สั่ง Modbus RS485 ทำงานในโหมด Slave RTU

< :mb485_mode 2 ; command echo

mbtcp_mode

กำหนดโหมดการทำงานของ Modbus บน TCP

Command :mbtcp_mode <mode>

; <mode> คือโหมดการทำงานมีดังนี้

; 0 = Disable, ปิดการทำงาน Modbus TCP

; 2 = Slave TCP

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :mbtcp_mode 2 ; สั่ง Modbus TCP ทำงานในโหมด Slave

< :mbtcp_mode 2 ; command echo

mb_addr

กำหนด Modbus Slave ID ระหว่าง 1-247

Command :mb_addr <id>

; <id> มีค่าระหว่าง 1-247 (1 คือค่าเริ่มต้นจากโรงงาน)

Response command echo (คืนอักษรเดิมที่ส่งไป)

mbtcp_port

กำหนดหมายเลข port สำหรับ Modbus TCP

port ของ Modbus TCP จะเป็นคนละ port ต่างจาก port หลักที่ใช้ในการสั่งงานหรืออ่านข้อมูล

Command :mbtcp_port <port>

; <port> คือหมายเลข port (ค่าเริ่มต้นจากโรงงานคือ 502)

Response command echo (คืนอักษรเดิมที่ส่งไป)

คำสั่งเกี่ยวกับเวลา

คำสั่งนี้ใช้สำหรับตั้งค่าหรืออ่านเวลาบนอุปกรณ์ DAQ-01S ที่มีนาฬิกาภายใน โดยจำเป็นในกรณีที่ต้องการสแกนข้อมูลพร้อม Timestamp หรือบันทึกข้อมูลลง SD card ภายในอุปกรณ์ หากอุปกรณ์เชื่อมต่ออินเทอร์เน็ตผ่านสาย LAN จะดึงเวลาอัตโนมัติจากอินเทอร์เน็ต ทำให้คำสั่งนี้ไม่จำเป็น

ymd

กำหนดปี, เดือน, วัน ให้กับอุปกรณ์

Command :ymd <yyyy/mm/dd>

; <yyyy/mm/dd> คือ ปีค.ศ. 4 หลัก / เดือน 2 หลัก / วัน 2 หลัก หากไม่ระบุ จะอ่านวันที่ในอุปกรณ์ออกมา

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

```
> :ymd 2024/01/23          ; ตั้งค่า ปี/เดือน/วัน
< :ymd 2024/01/23          ; command echo
> :ymd                     ; อ่านวันที่ในอุปกรณ์
< :ymd 2024/01/23          ; ค่าที่อ่านได้
```

hms

กำหนดเวลาชั่วโมง, นาที, วินาที ให้กับอุปกรณ์

Command :hms <hh:mm:ss>

; <hh:mm:ss> คือ ชั่วโมง 4 หลัก / เดือน 2 หลัก / วัน 2 หลัก หากไม่ระบุ จะอ่านวันที่ในอุปกรณ์ออกมา

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

```
> :hms 12:34:56            ; ตั้งค่า ชั่วโมง=12, นาที=34, วินาที=56
< :hms 12:34:56            ; command echo
> :hms                     ; อ่านเวลาในอุปกรณ์
< :hms 12:35:01            ; คือเวลาที่กำลังเดินในอุปกรณ์
```

คำสั่งอื่นๆ

getsw1

อ่านค่าจาก SW1 ว่ามีการกดหรือไม่

Command :getsw1

Response :getsw1 <val>

; <val> คือ ค่าที่อ่านได้

; 0 = กด

; 1 = ปล่อย

showled

แสดง RGB Led ให้เป็นสีที่ต้องการ เป็นระยะเวลาที่กำหนด

Command :showled

Response :showled <color> <period>

; <color> มีค่าระหว่าง 1-7 คือ หมายเลขสี มีดังนี้

; 1=red, 2=green, 3=yellow, 4=blue, 5=margenta, 6=cyan, 7=rgb(white)

; <period> มีค่าระหว่าง 1-300 คือระยะเวลาแสดง เป็นวินาที

setcounter

ตั้งค่าเริ่มต้นให้กับ counter

Command :setcounter <num> <val>

; <num> คือหมายเลข counter มีค่า 0,1

; <val> คือค่าเริ่มต้นที่ต้องการ

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :setcounter 0 2000 ; ตั้งค่า counter 0 = 2000

< :setcounter 0 2000 ; command echo

reset

ทำการ reset ค่าอุปกรณ์

Command :reset <type>

; <type> คือชนิดการ reset ขณะนี้มีค่าเดียวคือ

; 1 = reset Counter 0,1 คือ clear ค่า counter เป็น 0

Response command echo (คืนอักษรเดิมที่ส่งไป)

คำสั่งใช้งาน MQTT

mqtt_enable

กำหนดว่าให้ใช้งาน mqtt หรือไม่

Command :mqtt_enable <val>

; <val> มีดังนี้

; 0 = ปิด ไม่ทำงาน

; 1 = เปิด ให้ทำงาน

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :mqtt_enable 1 ; กำหนดให้ mqtt ทำงาน

< :mqtt_enable 1 ; command echo

➢ ***คำสั่งนี้ควรสั่งเมื่อตั้งค่า mqtt อื่นๆเรียบร้อยแล้ว จึงค่อยสั่งให้เริ่มต้นทำงาน***

mqtt_server

กำหนด url ของ mqtt server

Command :mqtt_server <url>

; <url> คือ url หรือหมายเลข IP ของ mqtt server (สูงสุด 60 ตัวอักษร)

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :mqtt_server mqtt.netpie.io ; กำหนด server mqtt ตาม NETPIE

< :mqtt_server mqtt.netpie.io ; command echo

mqtt_id, mqtt_user, mqtt_pswd

กำหนด mqtt device id, user และ password

Command :mqtt_id <id>

:mqtt_user <user>

:mqtt_pswd <pswd>

; <id>, <user>, <pswd> คือ device id, user และ password ตามที่กำหนดจาก server (ยาวสูงสุด 47 ตัวอักษร)

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :mqtt_id st4126789 ; กำหนด id= st4126789

< :mqtt_id st4126789 ; command echo

mqtt_topic

กำหนด topic ของ mqtt

Command :mqtt_topic <topic>

; <topic> คือ topic ตามที่ตกลงกับฝั่งรับ หรือ subscriber (สูงสุด 60 ตัวอักษร)

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :mqtt_topic @shadow/data/update ; กำหนด topic ตาม NETPIE สำหรับลง database

< :mqtt_topic @shadow/data/update ; command echo

mqtt_data_key, mqtt_ts_key

กำหนดชื่อ Key สำหรับ data และ timestamp ที่กำหนดมาจาก server

Command :mqtt_data_key <data_key>

:mqtt_ts_key <ts_key>

; <data_key> หรือ <ts_key> คือชื่อ Key ตามที่กำหนด

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :mqtt_data_key data ; กำหนดชื่อ data key ตาม NETPIE คือ data database

< :mqtt_data_key data ; command echo

mqtt_port

กำหนดหมายเลข port ของ mqtt เป็นค่าตามต้องการ (ค่าเริ่มต้นคือ 1883)

Command :mqtt_port <port>
; <port> คือ หมายเลข port ของ mqtt
Response command echo (คืนอักษรเดิมที่ส่งไป)
Example
> :mqtt_port 5678
< :mqtt_port 5678 ; command echo

mqtt_qos

กำหนดค่า QoS หรือ quality of service ที่ใช้ในการส่งข้อมูล

Command :mqtt_qos <qos>
; <qos> คือ ค่า qos มีค่า 0-2
Response command echo (คืนอักษรเดิมที่ส่งไป)
Example
> :mqtt_qos 1
< :mqtt_qos 1 ; command echo

mqtt_interval

กำหนดความเร็วหรืออัตราในการส่งข้อมูล เป็นคาบเวลา

Command :mqtt_interval <val>
; <val> คือ คาบเวลาหรือระยะห่างระหว่างข้อมูลที่ส่ง หน่วยเป็นวินาที ค่า 1-1200
Response command echo (คืนอักษรเดิมที่ส่งไป)
Example
> :mqtt_interval 5
< :mqtt_interval 5 ; command echo

mqtt_keepalive

กำหนดเวลา keep alive คือเวลาที่ใช้รักษาการเชื่อมต่อกับ mqtt broker ไว้ โดยต้องส่งคำสั่ง PINGREQ แทนหากไม่ส่งข้อมูลในระยะเวลาดังกล่าว

Command :mqtt_keepalive <val>
; <val> คือ เวลาในการ keep alive หน่วยเป็นวินาที ค่า 10-99
Response command echo (คืนอักษรเดิมที่ส่งไป)
Example
> :mqtt_keepalive 60
< :mqtt_keepalive 60 ; command echo

alists

กำหนดหมายเลขช่อง Analog Input ที่จะส่งข้อมูลด้วย mqtt โดยกำหนดเป็น 1 บิต 1 บิตต่อช่อง

Command :alists <val>

; <val> คือ ค่ารวมบิตของทุกช่อง Analog Input ที่จะส่ง เช่น 15 หมายถึงช่อง 0-3

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :alists 255

< :alists 255 ; command echo

dlists

กำหนดหมายเลขช่อง Digital Input ที่จะส่งข้อมูลด้วย mqtt โดยกำหนดเป็น 1 บิต 1 บิตต่อช่อง

Command :alists <val>

; <val> คือ ค่ารวมบิตของทุกช่อง Digital Input ที่จะส่ง ถ้าช่องไหนเป็น Counter หรือ Rate ก็จะส่งค่า Counter หรือ Rate แทน

Response command echo (คืนอักษรเดิมที่ส่งไป)

Example

> :dlists 255

< :dlists 255 ; command echo

การค้นหาอุปกรณ์

ก่อนจะเริ่มใช้งานอุปกรณ์ เราจะต้องค้นหาอุปกรณ์ว่าอยู่ที่ไหน แยกเป็น 2 กรณี

กรณีค้นหาผ่าน USB

การค้นหาผ่าน USB ต้องทำโดยการสแกนหา COM port โดยใช้ Function ที่แสดงรายการ COM port ที่มีอยู่ ในเครื่อง แล้วจึงทำการสแกนพอร์ตที่มีอยู่นั้นไปทีละพอร์ตด้วยการ

open COM port

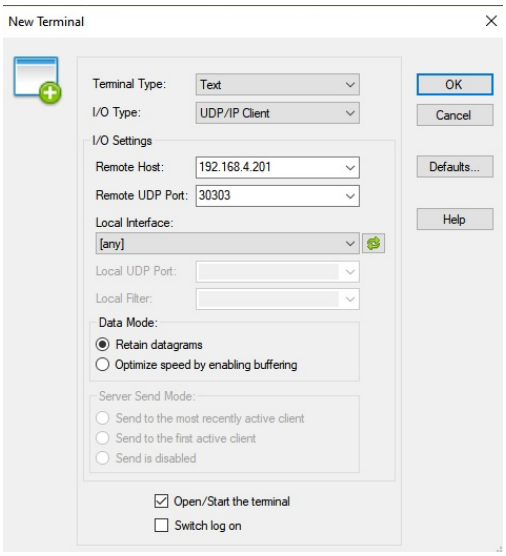
ส่งคำสั่งเช่น ":info 1<cr>" เพื่อให้กล่องตอบคืนค่า product name คือ "SmartBox" กลับมา หากไม่คืนในเวลาที่กำหนด ก็ close COM port

สแกนพอร์ตถัดไปจนกว่าจะพบ

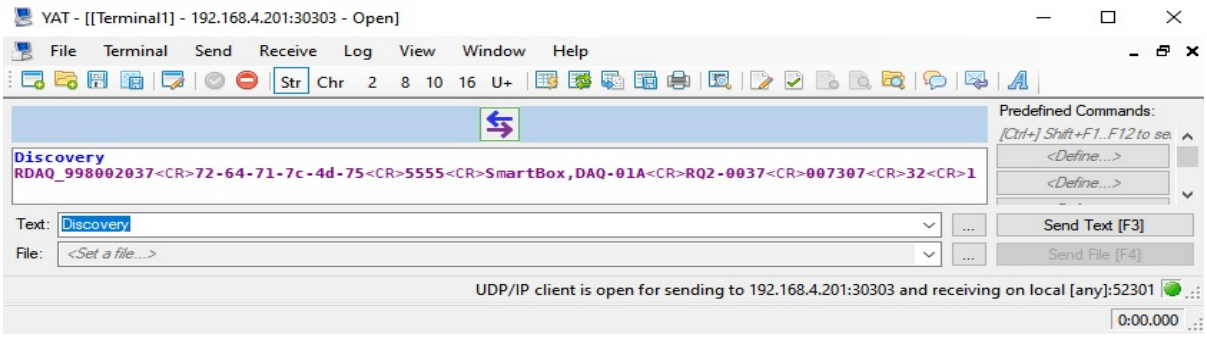
กรณีค้นหาผ่าน Ethernet

การค้นหาผ่าน Ethernet จะใช้วิธีค้นหาโดยการสแกนผ่าน IP โดยกำหนด Scan Range ที่ต้องการค้นหา แล้วเริ่มสแกนแต่ละ IP ด้วยการ ส่งคำสั่ง "Discovery" ด้วย UDP ไปทางพอร์ต 30303 แล้วรอดูว่าตอบมาหรือไม่ หากตอบมาจะได้ข้อมูลมา

ตัวอย่างเช่นใช้โปรแกรม YAT (ดาวน์โหลดได้ทาง Internet) เมื่อเริ่มโปรแกรม จะเปิดหน้าต่าง New Terminal ให้เลือกเปิดพอร์ตแบบ UDP โดยระบุ IP ของอุปกรณ์ และเปิดพอร์ต 30303 ดังรูป



ทำการส่งคำสั่ง “Discovery” ไปทางพอร์ตนี้ หากไม่ตอบกลับในเวลาที่กำหนด ก็ทำการสแกน IP ถัดไป หากมีข้อมูลจะตอบกลับมา จะได้ข้อมูลเกี่ยวกับ DAQ มีดังนี้



จากรูปจะได้ข้อมูลรวมกัน แยกเป็นส่วนๆด้วย <CR> เป็นรายการข้อมูลดังนี้

NetBIOS name : RDAQ_998002037 ชื่อ NetBIOS จะเป็น RDAQ_ ตามด้วย S/N ของอุปกรณ์

Mac address: 72-64-71-7c-4d-75 คือ mac address ของ Ethernet

Port: 5555 หมายเลขพอร์ต (command port)

Product_name, Model_name : SmartBox,DAQ-01A คือชื่อ Product ตามด้วย Model มี ‘,’คั่น

Device_name: RQ2-0037 คือชื่อ device name ที่ตั้งไว้

Firmware_version: 007307 คือรุ่นของ firmware = 0.73.07

ส่วน 2 ค่าสุดท้ายเป็น flag ของการทำงาน