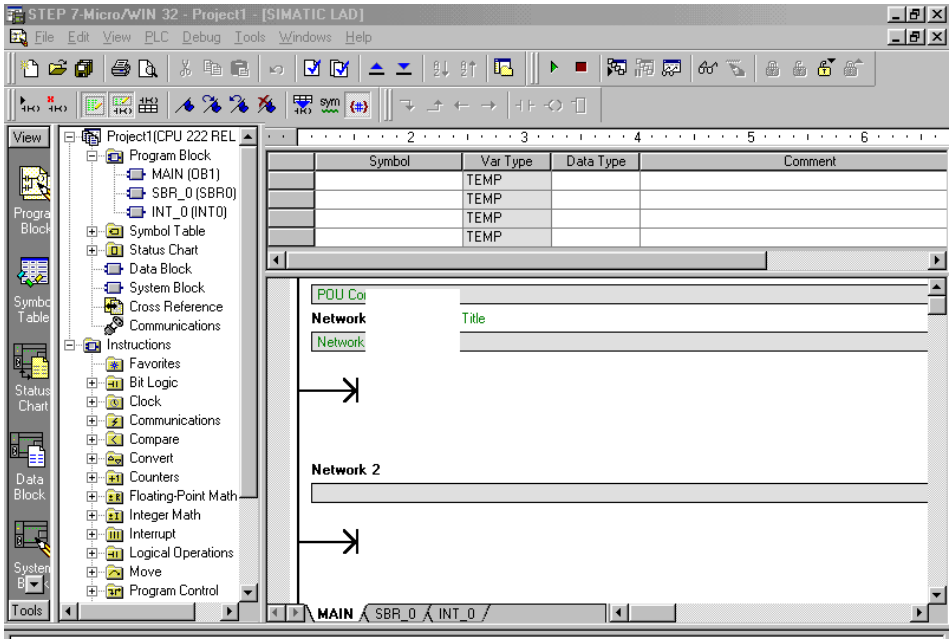
9.13 PLC SIEMENS S7-200

PLC S7-200 là một họ gồm nhiều loại CPU như CPU-212, 214, 215, 216, 224, 226. Các họ này khác nhau ở dung lượng nhớ, module I/O, tập lệnh, số cổng giao tiếp…, tuy nhiên về đại thể là giống nhau. PLC được lập trình thông qua cổng COM máy tính dùng chuẩn RS485 với phần mềm lập trình Step 7 Microwin ver 2.0 hay 3.x, 4.x theo kiểu kết nối PPI (point to point interface), nếu có cáp giao tiếp MPI (multi point interface) có thể ghép nối một PC với nhiều PLC. *Micro PLC Siemens ngoài họ S7-200 còn có họ S7-1200 lập trình bằng phần mềm Simatic Step7 Basic V10.5*

Chương trình PLC S7-200 được thiết kế dưới dạng chương trình chính (Main, OB), chương trình con (SBR), chương trình ngắt (INT), vùng nhớ dữ liệu (Data block), khối hệ thống (System Block)



S7-200 kết nối theo khối gồm khối CPU và các khối mở rộng, khối CPU có các ngõ vào ra số và cổng truyền thông RS485, cổng kết nối mạng. Số lượng ngõ vào ra số tuỳ loại CPU. Bộ nhớ gồm ba loại ROM, EEPROM và RAM và chia làm nhiều vùng: I, Q, AI, AQ, M, L, SM, T, C, V, HC, AC. Các ô nhớ có thể truy cập theo bit, byte (B), từ (W), từ kép (DW). ROM chứa hệ điều hành PLC, EEPROM là bộ nhớ thường trực, chứa chương trình, khối dữ liệu,

khối hệ thống, vùng nhớ V, nội dung hiện tại Timer TONR, nội dung hiện tại Counter, ô nhớ M nếu cài đặt cấu hình lưu giữ, RAM là vùng nhớ làm việc một phần được lưu trữ nhớ pin và siêu tụ. Khi cấp nguồn cho PLC nội dung trong EEPROM chuyển sang RAM để chạy chương trình.

Sau đây là bảng tóm tắt về các vùng nhớ:

Bảng 9.3

Miêu tả	CPU221	CPU222	CPU224/226
Chương trình người dùng	2KW	2KW	4KW
Dữ liệu người dùng (Data block)	1KW	1KW	2560W
Số module mở rộng	0	2	7
Ngõ vào số I (tối đa)	I0.0..I0.5	I0.0..I15.7	I0.0..I15.7
Ngõ vào số I (tích hợp trên module CPU)	6 I0.0..I0.5	8 I0.0..I0.7	14 I0.0..I0.7, I1.0..I1.5
Ngõ ra số Q (tối đa)	Q0.0..Q0.3	Q0.0..Q15.7	Q0.0..Q15.7
Ngõ ra số Q (tích hợp trên module (CPU)	4 Q0.0..Q0.3	6 Q0.0..Q0.5	10 Q0.0..Q0.7, Q1.0..1.1
Ngõ vào analog	AIW0..AIW30	AIW0..AIW30	AIW0..AIW30
Ngõ ra analog	AQW0..AQW30	AQW0..AQW30	AQW0..AQW30
Vùng nhớ thay đổi V, có thể giữ dữ liệu khi mất nguồn (retentive)	VB0..VB2047	VB0..VB2047	VB0..VB5519
Vùng nhớ trong M, có thể giữ dữ liệu khi mất nguồn tùy cài đặt	MB0..MB31	MB0..MB31	MB0..MB31
Vùng nhớ đặc biệt SM	SMB0..SMB179	SMB0..SMB299	SMB0..SMB549
Timer, có thể giữ dữ liệu khi mất nguồn	T0..T255	T0..T255	T0..T255
Counter, có thể giữ dữ liệu khi mất nguồn	C0..C255	C0..C255	C0..C255
Đếm vận tốc cao	HC0..HC3	HC0..HC3	HC0..HC5
Thanh ghi tích lũy ACC	AC0..AC3		
Vùng nhớ cục bộ L	LB0..LB59		
Vòng PID	8 vòng		

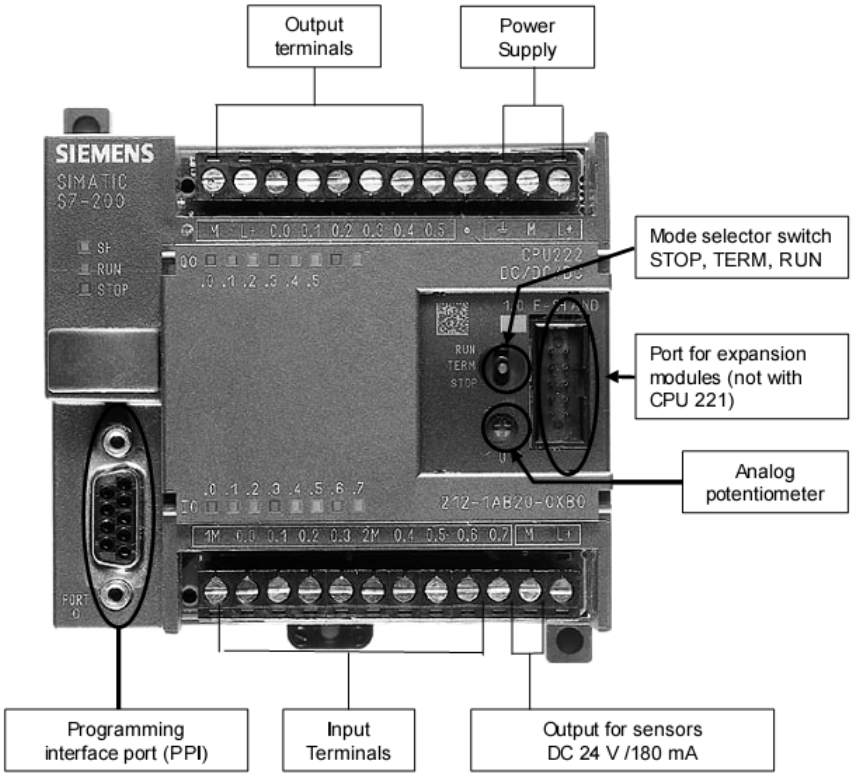
- Vùng nhớ AI, QI: chỉ truy cập theo từ (16 bit): AIW0, AQW10 dùng cho các ngõ vào/ra analog
- Vùng nhớ V, I, Q, M, SM: có thể truy cập theo bit, byte, từ hay từ kép: I0.1, QB2, VW150…

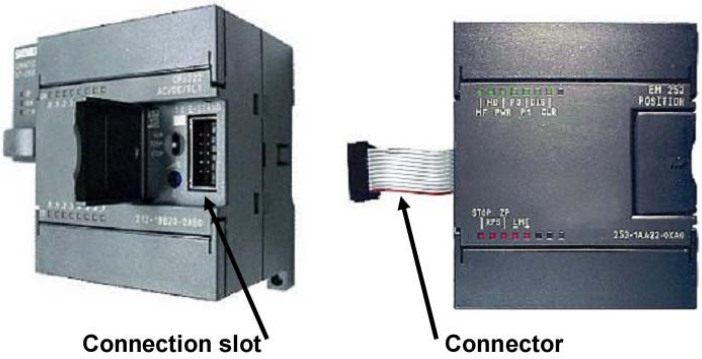
- T, C: truy cập theo bit hay từ: T1, C15 là trạng thái hay từ là nội dung 16 bit.
- AC: truy cập theo byte, từ hay từ kép .
- HC: truy cập theo từ kép.
- Vùng nhớ M, V, T, C có thể lưu lại khi mất điện.
- Một từ ví dụ VW0 gồm hai byte: VB100 (Byte cao) và VB101(Byte thấp)
- Một từ kép VD101 gồm hai từ: VW101 (Từ cao)và VW103, bốn byte VB100 (Byte cao), VB101, VB102, VB103 .
- Địa chỉ gián tiếp dùng ô nhớ 32 bit (VD, AC1,AC2, AC3, LD) làm con trỏ, giả sử AC1 là con trỏ, lệnh MOVD &VB0, AC1 đưa địa chỉ ô nhớ VB0 vào AC1, lệnh MOVW *AC1, MW 10 đưa nội dung ô nhớ VW0 sang ô nhớ MW10
- Số : số nguyên không dấu 8 bit (BYTE), 00..FF, 16 bit (WORD), 0000..FFFF, 32 bit (DWORD) . 00000000..FFFFFFFF, số có dấu mã phụ hai INT, DINT (số âm 80..7F, 6000..7FFF, 80000000..7FFFFFFF), số thực 32 bit REAL
- Hằng số thập phân 192, số thực +1.52E-2, 0.015, nhị phân 2#11000000, Hex 16#C0, ký tự ASCII 'AT', chuỗi ký tự "AT", chú ý là chuỗi và hằng đều biểu thị ký tự dạng mã ASCII nhưng ô nhớ đầu tiên của chuỗi là số ký tự trong chuỗi.

Các bit vùng SM cần lưu ý

SM 0.0	Luôn luôn ON
SM 0.1	ON ở chu kỳ quét đầu
SM 0.2	ON khi dữ liệu cần lưu trữ ở RAM bị mất (1 chu kỳ)
SM 0.3	ON khi RUN
SM 0.4	Xung nhịp chu kỳ 1 phút
SM 0.5	Xung nhịp chu kỳ 1 sec
SM 0.6	Xung nhịp có chu kỳ bằng hai lần chu kỳ quét
SM 0.7	Phản ảnh vị trí MODE SWITCH của PLC, OFF:TERM, ON :RUN
SM 1.0	ON khi kết quả tính là Zero
SM 1.1	ON khi bị tràn
SM 1.2	ON khi kết quả âm
SM 1.3	ON khi chia cho zero
SM 1.4	ON khi bảng bị tràn (xem lệnh bảng)
SM 1.5	ON khi bảng bị trống (xem lệnh bảng)
SM 1.6	ON khi lệnh BCD-I không thực hiện được
SM 1.7	ON khi lệnh ATH không thực hiện được
SMB2	Chứa ký tự nhận từ Port 0/1 ở chế độ truyền thông freeport
SM3.1	ON khi sai parity Port 0/1
SMB28	Thay đổi tùy theo vị trí biến trở analog 1
SMB29	Thay đổi tùy theo vị trí biến trở analog 2

Để mở rộng khả năng ta ghép thêm các module mở rộng, tối đa 7 module cho CPU 224/226

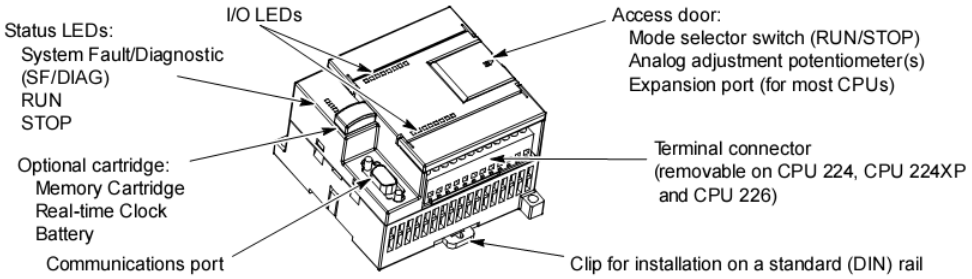


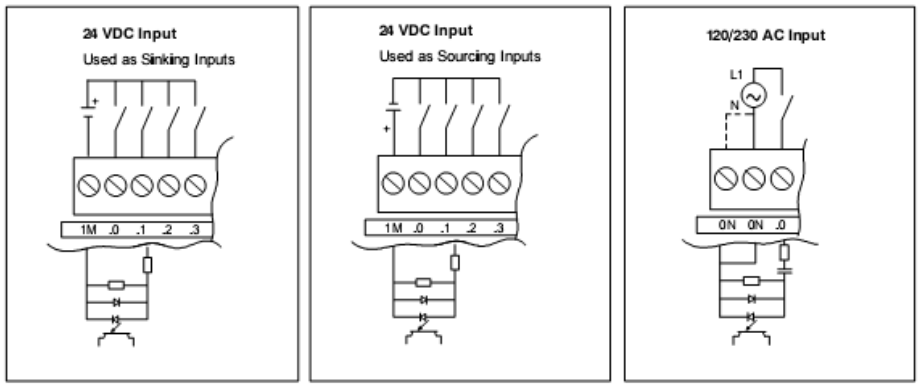


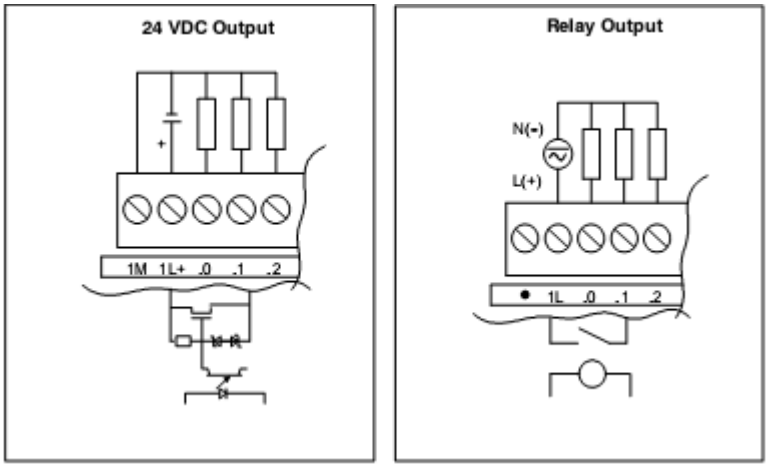
THÔNG SỐ CÁC MODULE

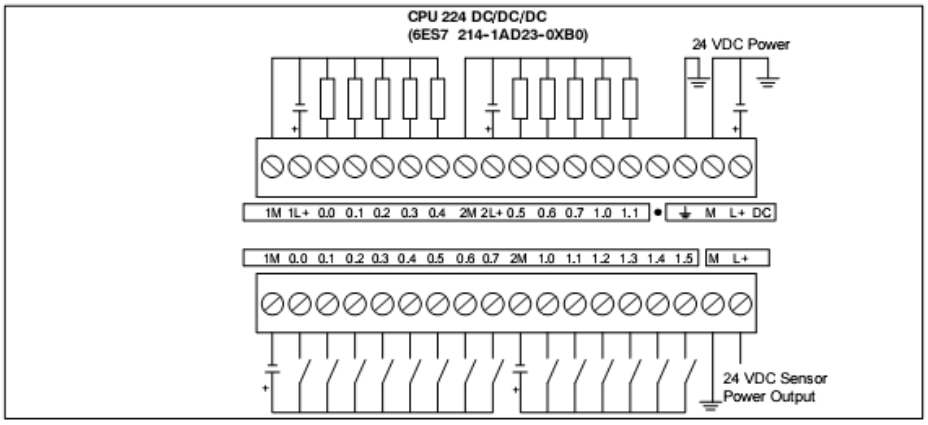
a/ MODULE CPU

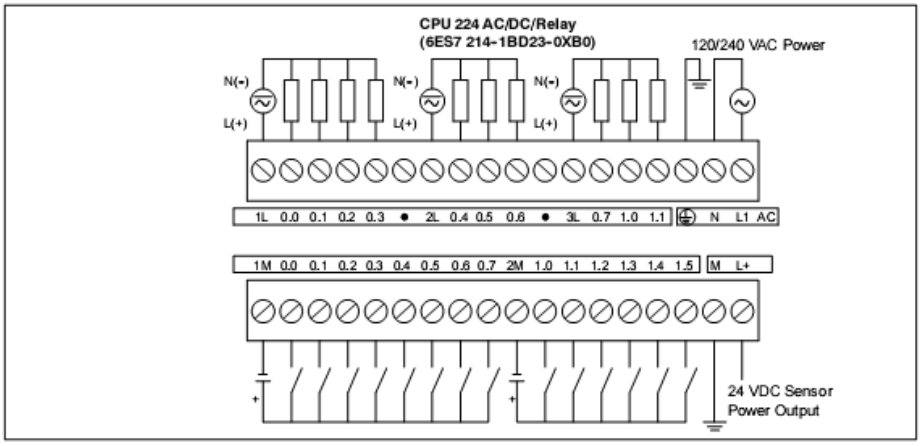
	CPU224	**CPU224XP**	**CPU226**
Cấp nguồn	24VDC hay 120 đến 240VAC		
Vào/ra số	14 vào 24VDC. 10 ra 24VDC hay relay		24 vào 24VDC. 16 ra 24VDC hay relay
Số lượng	tối đa 128 vào, 128 ra		
Vào/ra analog	Không có	2 vào, 1 ra	Không có
Số lượng	Tối đa 32 vào, 32 ra		
Số module mở rộng tối đa	7		
Bộ đếm tần số cao	6		
Phát xung tần số cao	2		
Thực hiện lệnh boolean	0.22 µsec		
RTC	Có		
Cartridge	Thẻ nhớ và pin		
Port	1 RS485	2 RS485	
Chương trình người dùng	8KB	12KB	16KB
Data	8KB	10KB	10KB
Backup (super capacitor)	100 giờ		
Pin	100 ngày		


I/O LEDs
Status LEDs:
System Fault/Diagnostic (SF/DIAG)
RUN
STOP
Optional cartridge:
Memory Cartridge
Real-time Clock
Battery
Communications port
Access door:
Mode selector switch (RUN/STOP)
Analog adjustment potentiometer(s)
Expansion port (for most CPUs)
Terminal connector
(removable on CPU 224, CPU 224XP and CPU 226)
Clip for installation on a standard (DIN) rail


24 VDC Input
Used as Sinking Inputs
1M .0 .1 .2 .3
24 VDC Input
Used as Sourcing Inputs
1M .0 .1 .2 .3
120/230 AC Input
L1
N
0N 0N .0


24 VDC Output
1M 1L+ .0 .1 .2
Relay Output
N(-)
L(+)
1L .0 .1 .2

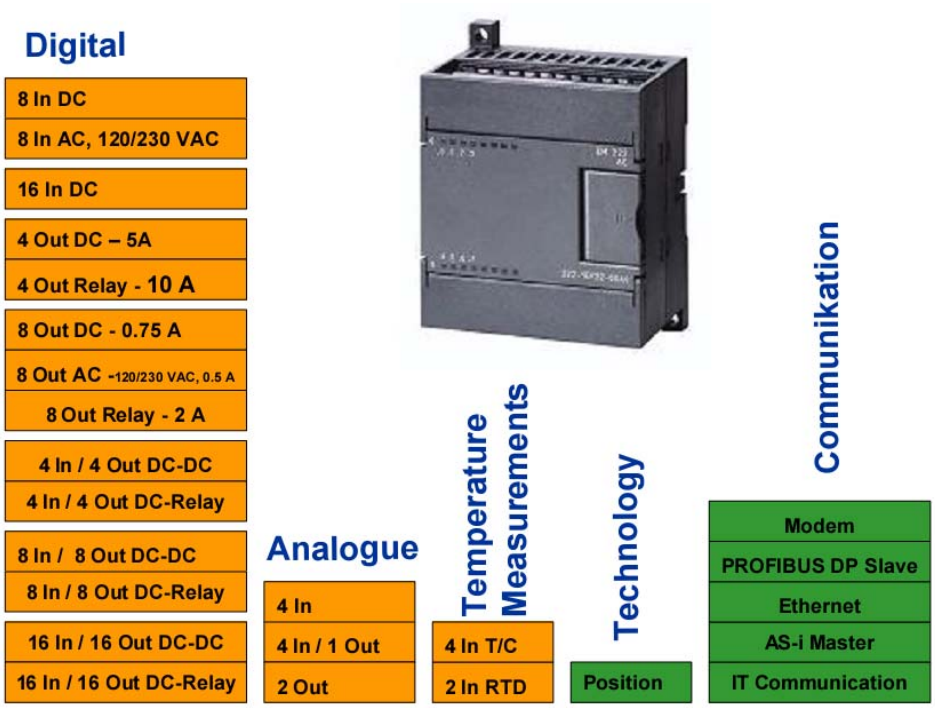




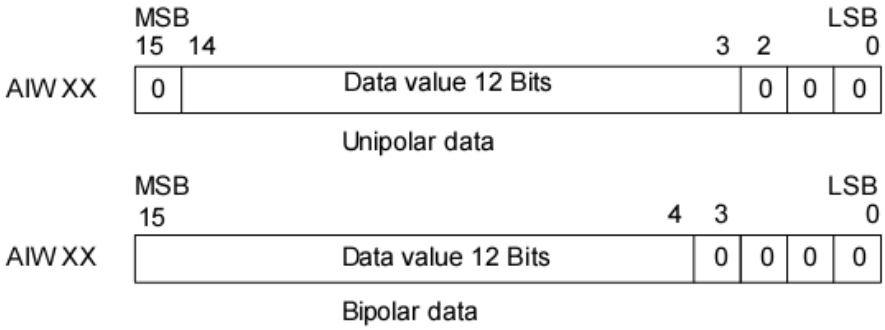
b/ Các module mở rộng

EM221	Digital Input
EM222	Digital Output
EM223	Digital Input/Output
EM231	Analog Input
EM232	Analog Output
EM235	Analog Input/Output
EM231	Thermocouple Input
EM231	RTD Input
EM277	Profibus- DP module
EM241	Modem module
EM253	Position module
CP 243-1	Ethernet module

CP 243-1 IT	Internet module
CP 243-2	ASI module



Ta cài đặt cấu hình cho các module analog thông qua switch cấu hình gắn trên module, chuyển đổi ADC có phân giải 12 bit còn DAC có phân giải 11 bit, do đó các bit thấp trong ô nhớ AIW, AQW sẽ không có giá trị, giá trị analog đơn cực đi từ 0 đến 32000, lưỡng cực đi từ -32000 đến +32000



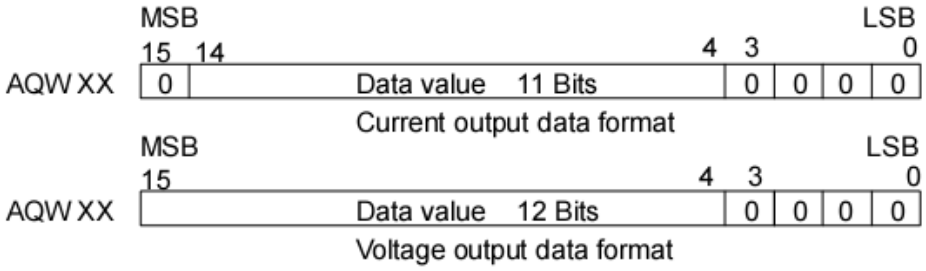


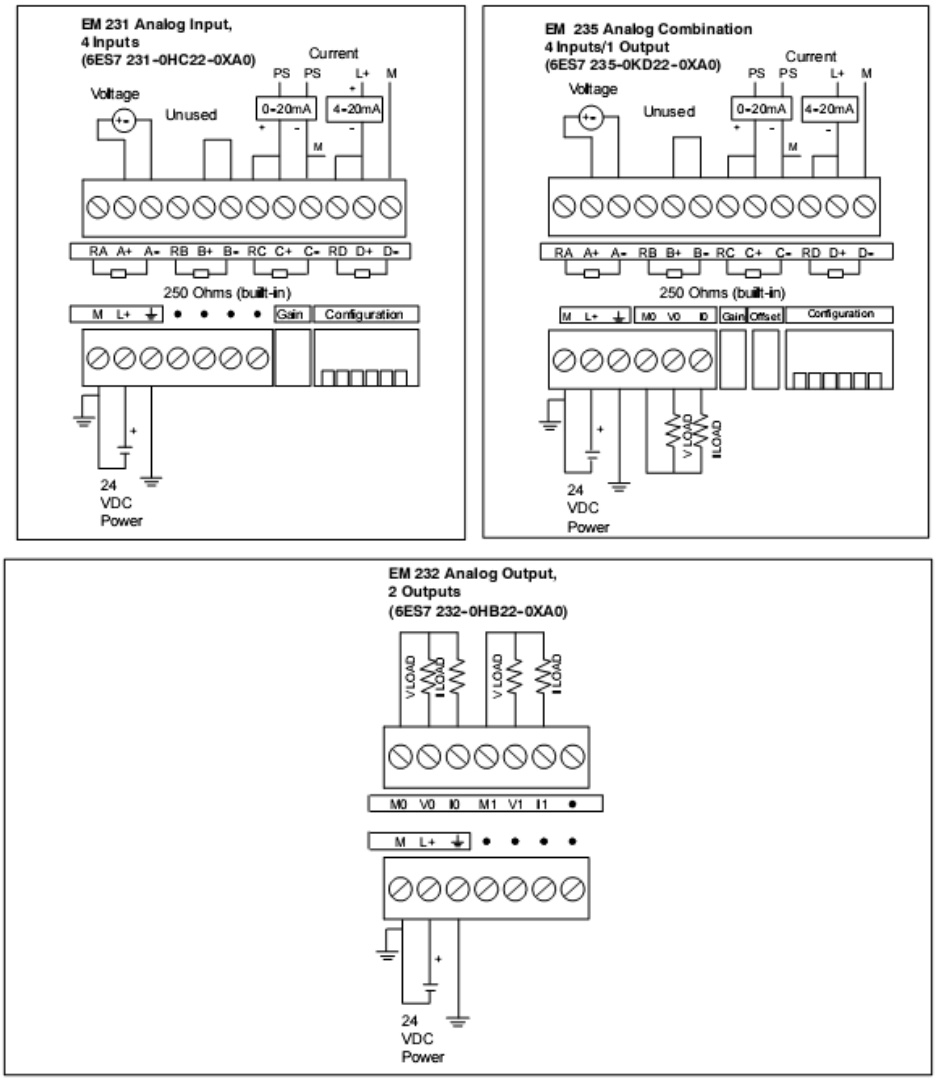
Table A-20 EM 231 Configuration Switch Table to Select Analog Input Range

Unipolar			Full-Scale Input	Resolution
SW1	SW2	SW3		
ON	OFF	ON	0 to 10 V	2.5 mV
	ON	OFF	0 to 5 V	1.25 mV
			0 to 20 mA	5 μA
Bipolar			**Full-Scale Input**	**Resolution**
SW1	**SW2**	**SW3**		
OFF	OFF	ON	±5 V	2.5 mV
	ON	OFF	± 2.5 V	1.25 mV

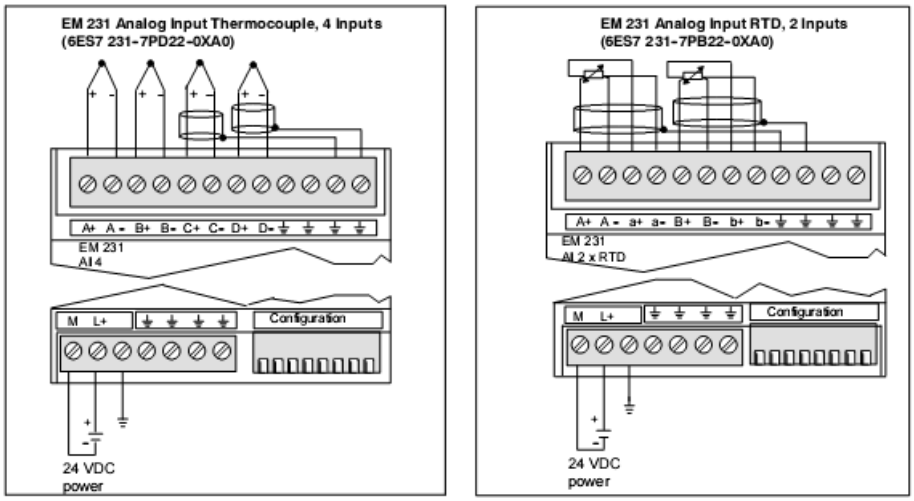
Table A-21 EM 235 Configuration Switch Table to Select Analog Range and Resolution

Unipolar						Full-Scale Input	Resolution
SW1	SW2	SW3	SW4	SW5	SW6		
ON	OFF	OFF	ON	OFF	ON	0 to 50 mV	12.5 μV
OFF	ON	OFF	ON	OFF	ON	0 to 100 mV	25 μV
ON	OFF	OFF	OFF	ON	ON	0 to 500 mV	125 μV
OFF	ON	OFF	OFF	ON	ON	0 to 1 V	250 μV
ON	OFF	OFF	OFF	OFF	ON	0 to 5 V	1.25 mV
ON	OFF	OFF	OFF	OFF	ON	0 to 20 mA	5 μA
OFF	ON	OFF	OFF	OFF	ON	0 to 10 V	2.5 mV

Bipolar						Full-Scale Input	Resolution
SW1	SW2	SW3	SW4	SW5	SW6		
ON	OFF	OFF	ON	OFF	OFF	±25 mV	12.5 μV
OFF	ON	OFF	ON	OFF	OFF	±50 mV	25 μV
OFF	OFF	ON	ON	OFF	OFF	±100 mV	50 μV
ON	OFF	OFF	OFF	ON	OFF	±250 mV	125 μV
OFF	ON	OFF	OFF	ON	OFF	±500 mV	250 μV
OFF	OFF	ON	OFF	ON	OFF	±1 V	500 μV
ON	OFF	OFF	OFF	OFF	OFF	±2.5 V	1.25 mV
OFF	ON	OFF	OFF	OFF	OFF	±5 V	2.5 mV
OFF	OFF	ON	OFF	OFF	OFF	±10 V	5 mV



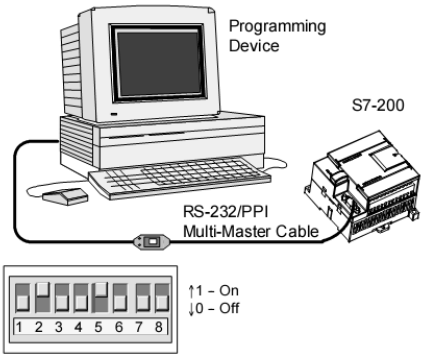
Các module Thermocouple và RTD đo nhiệt độ với phân giải 0,1 độ, giá trị đọc là nhiệt độ nhân với 10, các switch cấu hình chọn loại cảm biến, thang nhiệt độ…



9.14 CÁC BƯỚC LẬP TRÌNH

- Cấp nguồn PLC

- Nối cáp PLC và máy tính, đặt switch trên cáp vị trí 01001000 (9600, E, 8,1)

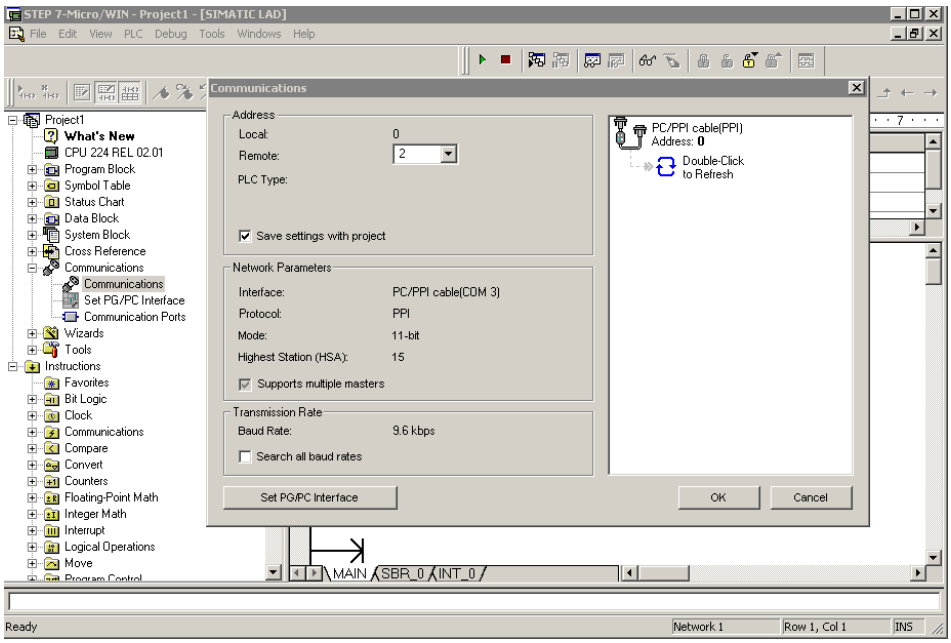


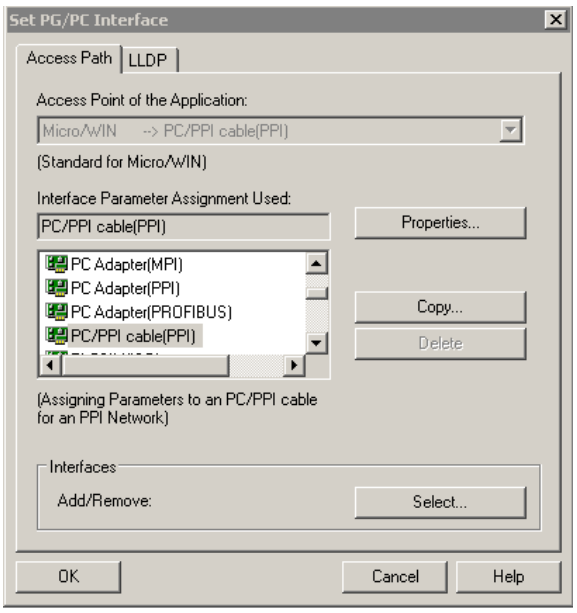
- Chạy chương trình Microwin, tạo Project và kết nối với PLC: kích chuột vào Communications hiện cửa sổ Communications, địa chỉ máy tính là 0, địa chỉ PLC là 2, kích chuột vào Double Click to Refresh, chương trình sẽ đi tìm PLC đang kết nối và hiển thị loại PLC, các module mở rộng đang kết nối; nếu không kết nối được, vào Set PC/PG interface cài đặt thông số truyền thông cho phù hợp với thông số trên cáp.

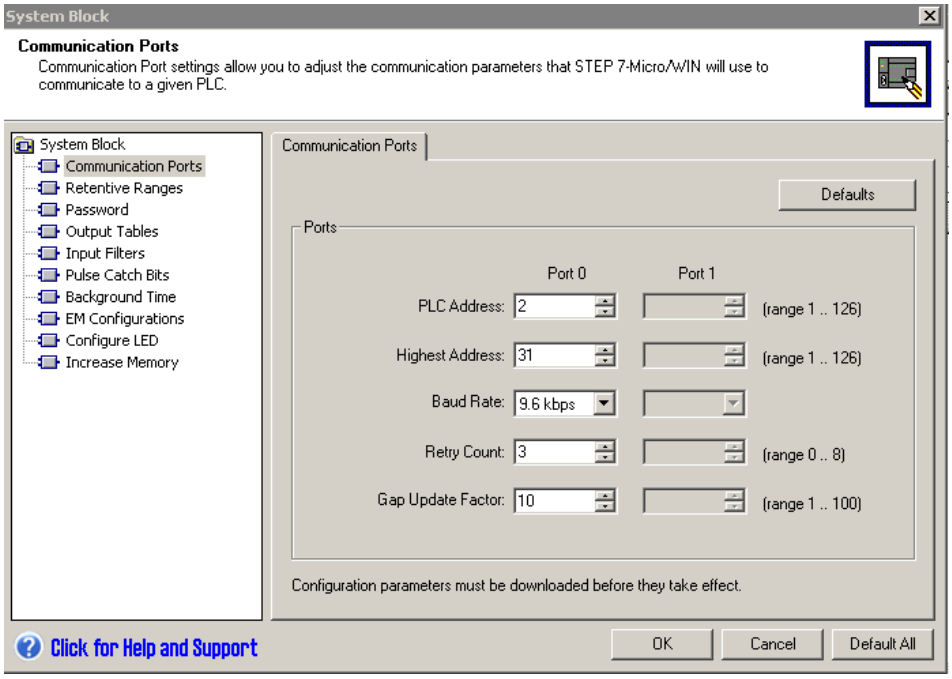
- Cài đặt cấu hình cho PLC, vào System Blocks cài đặt cổng truyền thông (địa chỉ PLC khi nối mạng, vận tốc truyền 9600 b/s,19200 b/s, 187,5 kb/s), vùng nhớ lưu giữ, password, lọc ngõ vào, cài trạng thái ngõ ra số khi PLC chuyển sang Stop…Cấu hình này phải download xuống PLC mới có hiệu lực .

- Lập trình, compile rồi download chương trình hay/và khối hệ thống, khối dữ liệu xuống PLC

- Chuyển PLC sang RUN và kiểm tra hoạt động: Debug-Start Program Status

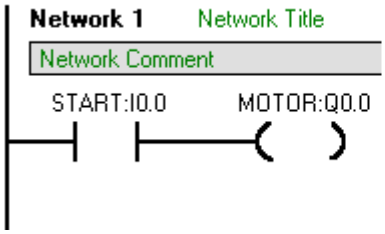



Set PG/PC Interface
Access Path
LLDP
Access Point of the Application:
Micro/WIN --> PC/PPI cable(PPI)
(Standard for Micro/WIN)
Interface Parameter Assignment Used:
PC/PPI cable(PPI)
Properties...
PC Adapter(MPI)
PC Adapter(PPI)
PC Adapter(PROFIBUS)
PC/PPI cable(PPI)
Copy...
Delete
(Assigning Parameters to an PC/PPI cable for an PPI Network)
Interfaces
Add/Remove:
Select...
OK
Cancel
Help

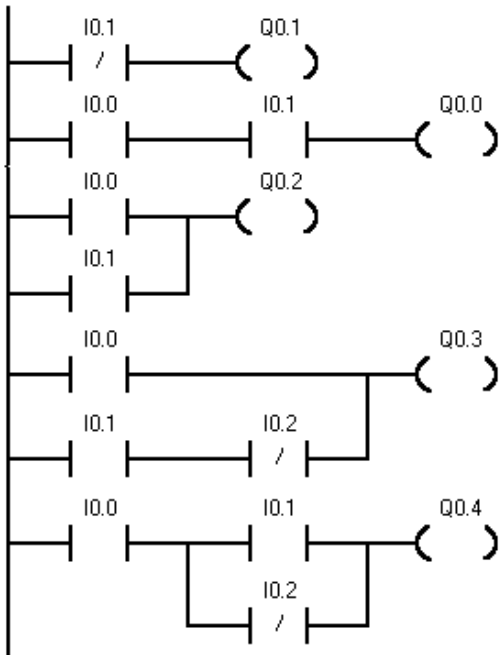

System Block
Communication Ports
Communication Port settings allow you to adjust the communication parameters that STEP 7-Micro/WIN will use to communicate to a given PLC.
System Block
Communication Ports
Retentive Ranges
Password
Output Tables
Input Filters
Pulse Catch Bits
Background Time
EM Configurations
Configure LED
Increase Memory
Communication Ports
Defaults
Ports
Port 0
Port 1
PLC Address: 2 (range 1 .. 126)
Highest Address: 31 (range 1 .. 126)
Baud Rate: 9.6 kbps
Retry Count: 3 (range 0 .. 8)
Gap Update Factor: 10 (range 1 .. 100)
Configuration parameters must be downloaded before they take effect.
Click for Help and Support
OK
Cancel
Default All

9.15. LỆNH CƠ BẢN

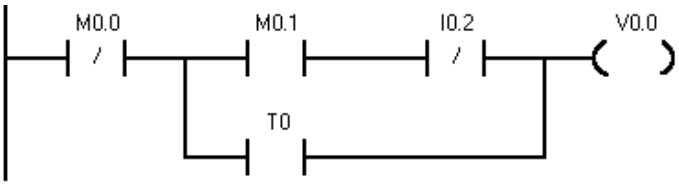
9.15.1 Lệnh bit



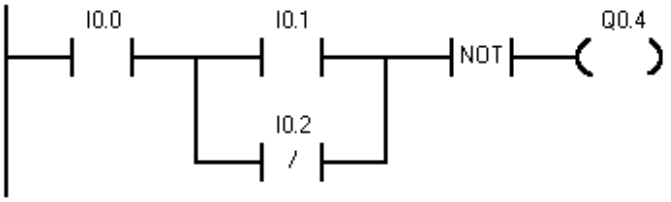

```
Network 1      Network Title
Network Comment
LD      START:I0.0
=       MOTOR:Q0.0
```



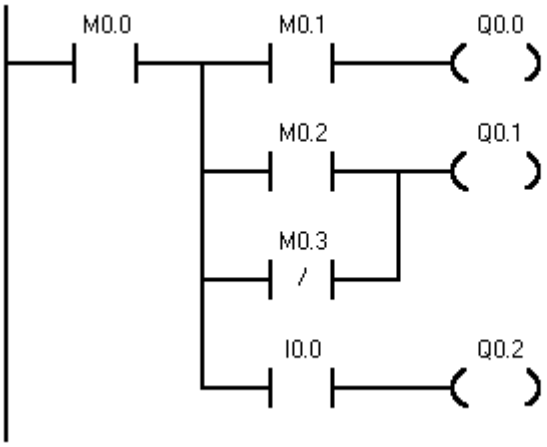

```
LDN     I0.1
=       Q0.1
LD      I0.0
A       I0.1
=       Q0.0
LD      I0.0
O       I0.1
=       Q0.2
LD      I0.0
LD      I0.1
AN      I0.2
OLD
=       Q0.3
LD      I0.0
LD      I0.1
ON      I0.2
ALD
=       Q0.4
```




```
LDN     M0.0
LD      M0.1
AN      I0.2
O       T0
ALD
=       V0.0
```

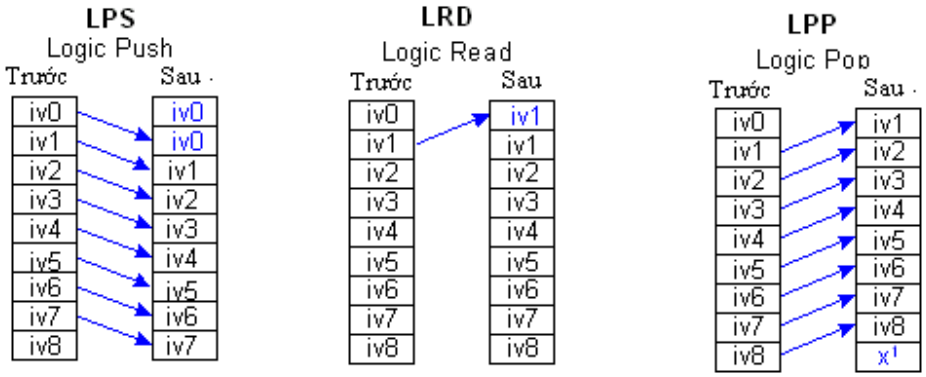



```
LD      I0.0
LD      I0.1
ON      I0.2
ALD
NOT
=       Q0.4
```

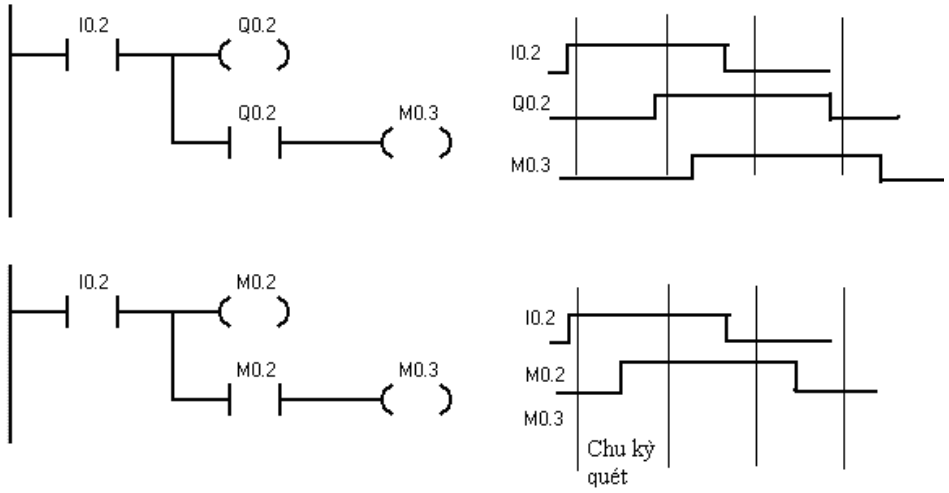



```
LD      M0.0
LPS
A       M0.1
=       Q0.0
LRD
LD      M0.2
ON      M0.3
ALD
=       Q0.1
LPP
A       I0.0
=       Q0.2
```

LD (Load), LDN (Load Not) là lệnh bắt đầu một network hay một nhóm điều kiện, ALD (And Load) dùng ghép nối tiếp hai nhóm, OLD (Or Load) ghép song song hai nhóm. Khi có rẽ nhánh các điều kiện ta dùng các lệnh ngăn xếp để cất hay lấy ra kết quả logic ở điểm rẽ. Có ba lệnh ngăn xếp thường sử dụng : LPS (logic push), LPP (logic pop) và LRD (logic read)

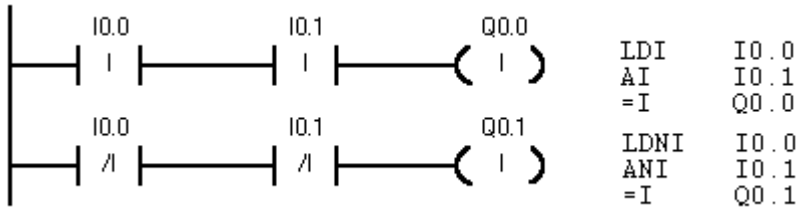


Chú ý là các ngõ ra Q chỉ thay đổi giá trị ở cuối chu kỳ quét. So sánh hai network sau



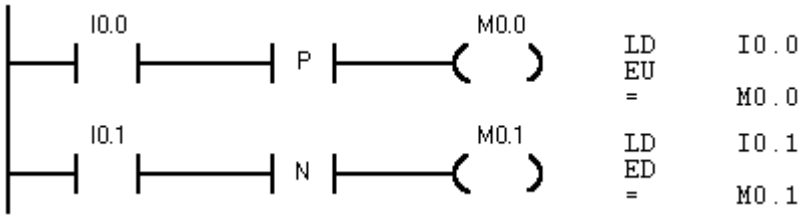
9.15.2 Lệnh lập tức

Lệnh lập tức ký hiệu bằng chữ I, bit nhớ trong sẽ phản ảnh ngay ngõ vào I không chờ đến đầu chu kỳ quét, tương tự bit nhớ trong xuất ngay ra ngõ xuất Q



9.15.3 Lệnh vi phân

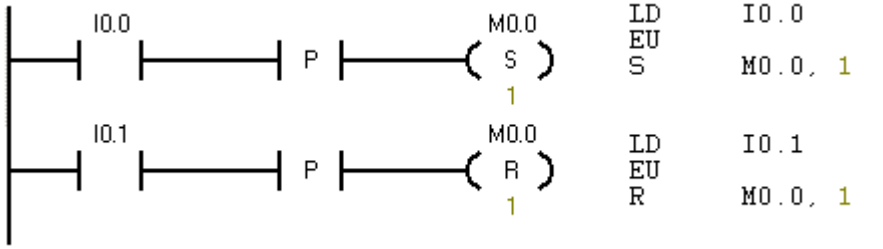
Lệnh này xét sự chuyển trạng thái của điều kiện từ OFF sang ON (vi phân cạnh lên) hay ON sang OFF (vi phân cạnh xuống) logic sau lệnh sẽ ON trong một chu kỳ quét

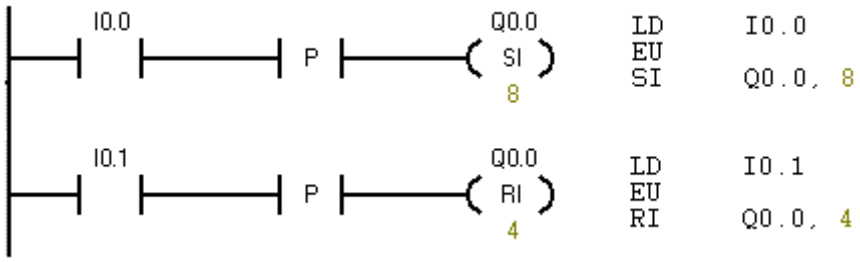


9.15.4 Lệnh đặt/ xoá

Lệnh S, SI cho một loạt n bit (1..255) liên tiếp nhau ON khi điều kiện ON và giữ nguyên khi điều kiện OFF trở lại

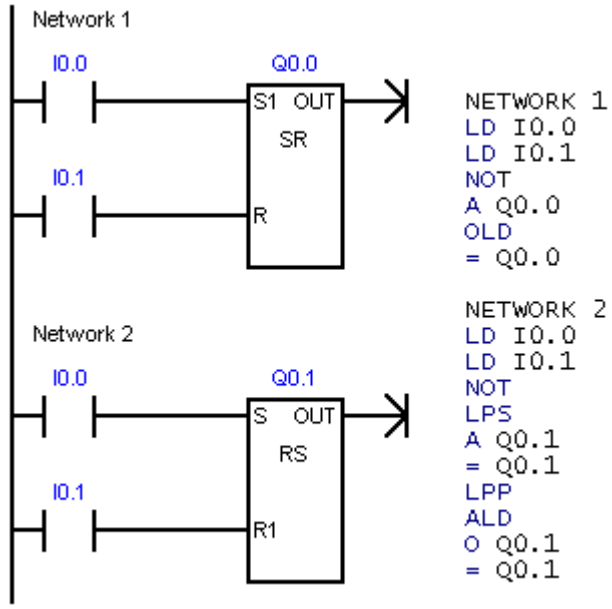
Lệnh R, RI cho một loạt n bit liên tiếp nhau OFF khi điều kiện ON và giữ nguyên khi điều kiện OFF trở lại

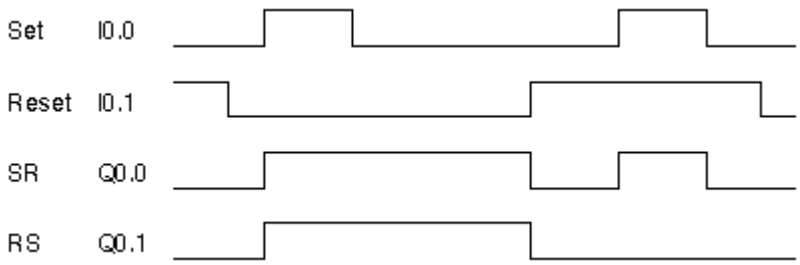




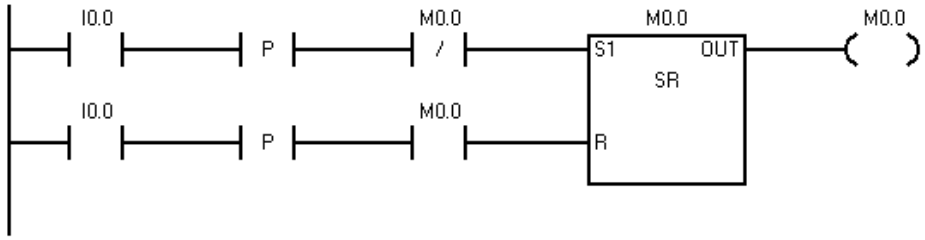
Lệnh SR có hai ngõ vào S và R, nếu S ON và R ON đồng thời hay OFF thì một bit được set, nếu S OFF và R ON thì bit được xóa, ngõ vào S ưu tiên hơn

Lệnh RS có hai ngõ vào S và R, nếu R ON và S ON đồng thời hay OFF thì một bit được xóa, nếu R OFF và S ON thì bit được set, ngõ vào R ưu tiên hơn

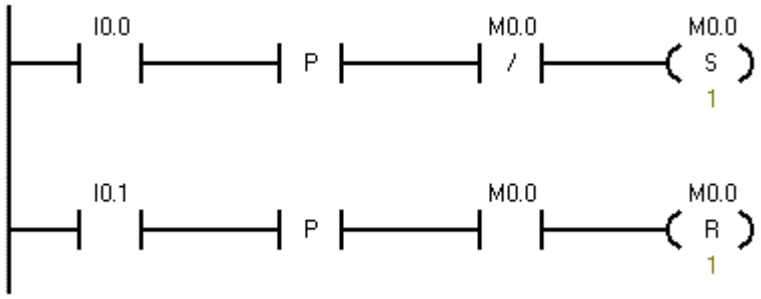




Ví dụ: bấm I0.0 thì M0.0 được set, bấm lần nữa thì M0.0 bị xóa



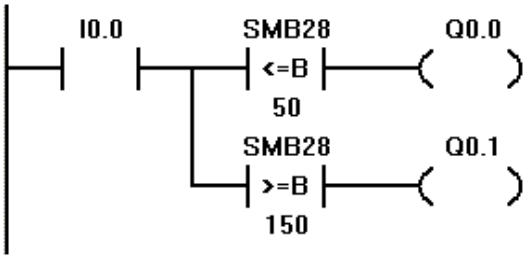
Tuy nhiên sơ đồ dưới không thực hiện được ý đồ này



9.16 LỆNH SO SÁNH

So sánh hai số n1 và n2 theo byte không dấu B, số nguyên có dấu 16 bit I, số nguyên có dấu 32 bit D, số thực R theo các phép <, ==, >, <>, <=, >=, kết quả so sánh là logic ON nếu thỏa điều kiện so sánh.

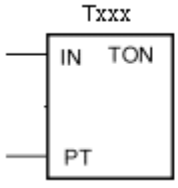
Ví dụ:




```
LD      I0.0
LPS
AB<=    SMB28, 50
=       Q0.0
LPP
AB>=    SMB28, 150
=       Q0.1
```

9.17 LỆNH TIMER/COUNTER

Có ba loại timer là ON delay, Retentive ON delay và OFF delay. Thời gian timer bằng giá trị PT nhân hệ số tùy thuộc xxx.



Khi IN là ON sau thời gian trễ tiếp điểm TXX sẽ ON, nội dung bộ đếm tiếp tục tăng đến tối đa 32767.

Khi IN là OFF thì tiếp điểm là OFF và nội dung bộ đếm bằng 0.

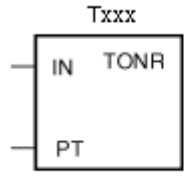
Hệ số 1ms : T32, T96

10ms : T33..T36, T97..T100

100ms: T37..T63, T101..T255

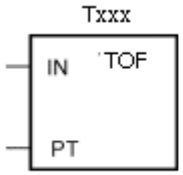
PT: VW, T, C, IW, QW, MW, SMW, SW, AC, AIW, Hằng số, *VD, *AC

PT tối đa 32767



TONR khác TON ở chỗ khi IN là OFF thì nội dung timer không bị xóa, bộ đếm tiếp tục tăng khi IN từ OFF sang ON cho đến khi nội dung timer bằng PT thì tiếp điểm Txxx đóng. Bộ đếm tiếp tục đếm đến 32767. Muốn xóa Txxx ta phải dùng lệnh reset R

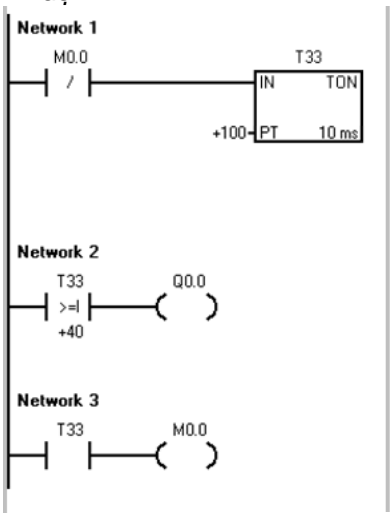
Hệ số 1ms : T0, T64
10ms : T1..T4, T65..T68
100ms: T5..T31, T69..T95



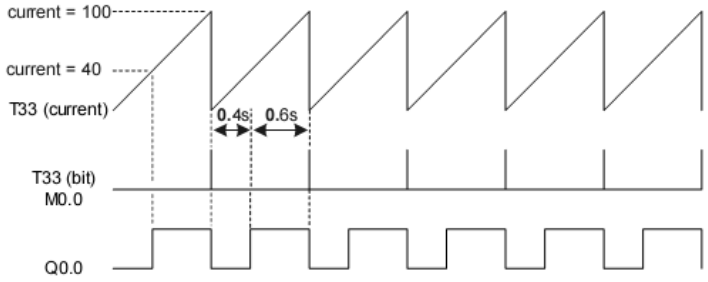
Khi IN ON thì Txx ON, bộ đếm xoá, khi IN OFF bộ đếm bắt đầu đếm lên, khi bằng PT thì Txxx OFF và ngừng đếm
Nếu thời gian IN OFF ngắn hơn delay thì Txxx vẫn ON

Hệ số TOF giống TON, không được dùng TON và TOF cùng số.

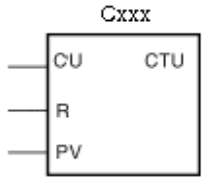
Ví dụ:




```
Network 1       //10 ms timer T33 times out after
                //(100 x 10 ms = 1s)
                //M0.0 pulse is too fast to monitor
                //with Status view
LDN     M0.0
TON     T33, +100
Network 2       //Comparison becomes true at a
                //rate that is visible with Status view.
                //Turn on Q0.0 after (40 x 10 ms)
                //for a 40% OFF/60% ON waveform
LDW>=   T33, +40
=       Q0.0
Network 3       //T33 (bit) pulse too fast to monitor
                //with Status view
                //Reset the timer through M0.0 after
                //the (100 x 10 ms)  period
LD      T33
=       M0.0
```



Có ba loại đếm: đếm tăng CTU, đếm giảm CTD và đếm tăng giảm CTUD

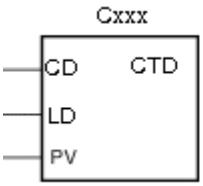


Khi R là ON thì bộ đếm bị xóa

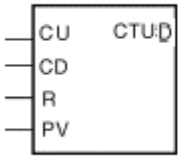
Khi có xung từ OFF sang ON vào ngõ CU thì bộ đếm tăng 1

Khi nội dung bộ đếm bằng PV thì tiếp điểm Cxxx là ON, nội dung bộ đếm tiếp tục tăng theo IN đến tối đa 32767

xxx: 0..255



LD ON nạp PV vào bộ đếm và xoá Cxxx
LD OFF , bộ đếm giảm khi CD chuyển từ OFF sang ON
Khi giảm đến 0 Cxxx ON, bộ đếm ngừng đếm



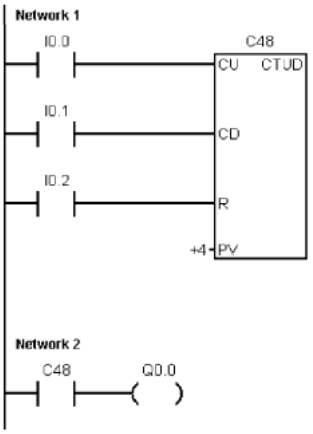
Bộ đếm sẽ tăng hoặc giảm tùy xung vào CU hay CD và khi R OFF

Khi R ON nội dung bộ đếm là 0, Cxxx OFF

khi nội dung bộ đếm >= PV, Cxxx ON

xxx: 0 đến 255

Ví dụ :



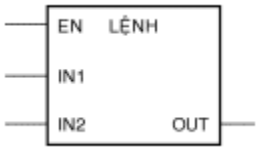

```
Network 1       //I0.0 counts up
                //I0.1 counts down
                //I0.2 resets current value to 0
LD      I0.0
LD      I0.1
LD      I0.2
CTUD    C48, +4

Network 2       //Count Up/Down counter C48
                //turns on C48 bit  when
                //current value >= 4
LD      C48
=       Q0.0
```

9.18 LỆNH SỐ HỌC

- ADD_I/ SUB_I/ MUL_I/ DIV_I Cộng/ trừ/ nhân/ chia số nguyên 16 bit in1 và in2, kết quả trong out
- ADD_DI/ SUB_DI/ MUL_DI/ DIV_DI Cộng/ trừ/ nhân/ chia

số nguyên 32 bit, kết quả 32 bit.

- ADD_R/ SUB_R/ MUL_R/ DIV_R Cộng/ trừ/ nhân/ chia số thực 32 bit



LỆNH: mã lệnh

IN1, IN2: toán hạng

OUT: nơi chứa kết quả

Trong STL các lệnh này có dạng *lệnh in1, out*

+I, +D, +R: out= in1 + out

-I, -D, -R : out= out - in1

*I, *D, *R : out= out * in1

/I, /D, /R : out= out/ in1

Để ý là

- MUL_I: nhân hai số 16 bit, kết quả 16 bit
- DIV_I: chia hai số 16 bit, kết quả 16 bit

Còn hai lệnh đặc biệt

- MUL: nhân hai số 16 bit, kết quả 32 bit
- DIV: chia hai số 16 bit cho thương số 16 bit trong word thấp và dư số 16 bit trong word cao của từ kép out

Trong STL các lệnh này có dạng

MUL in1, out: nhân word thấp của out với word in1, kết quả trong out

DIV in1, out: word thấp của out chia cho word in1, dư số 16 bit trong word cao của từ kép out, thương số 16 bit trong word thấp

- Căn bậc hai số thực SQRT
- SIN, COS, TAN, EXP, LN
- Tăng/ giảm ô nhớ byte, word, double word in và chứa kết quả ra out INC_X/ DEC_X X=B, W, D

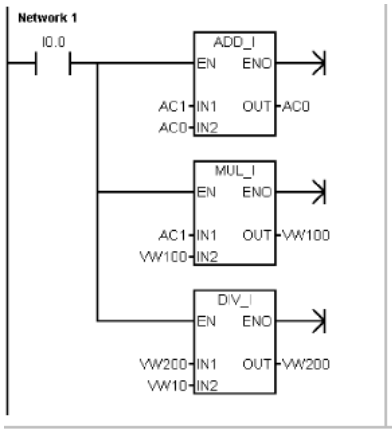
STL: INCX/DECX out, tăng giảm out và đưa kết quả ra out

Khi phép tính được thực hiện có lỗi thì ENO= 0, các ô nhớ không đổi giá trị, các lỗi gồm có : tràn SM1.1 = 1; chia cho zero SM1.3=1.

Nếu kết quả là zero, SM1.0=1

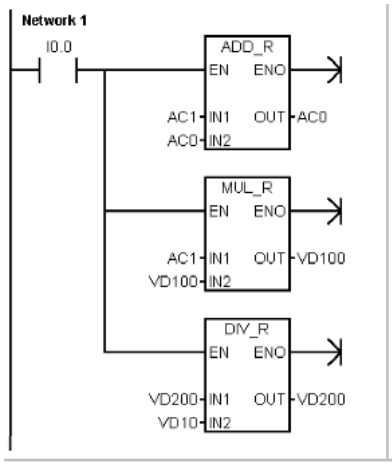
Nếu kết quả âm, SM1.2=1

Ví dụ:



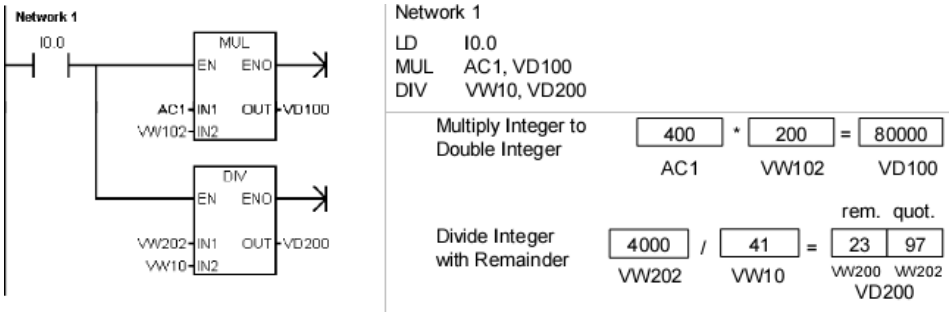

```
Network 1
LD    I0.0
+I    AC1, AC0
*I    AC1, VW100
/I    VW10, VW200
```

Add				Multiply				Divide			
40	+ 60	= 100		40	* 20	= 800		4000	/ 40	= 100	
AC1	AC0	AC0		AC1	VW100	VW100		VW200	VW10	VW200	



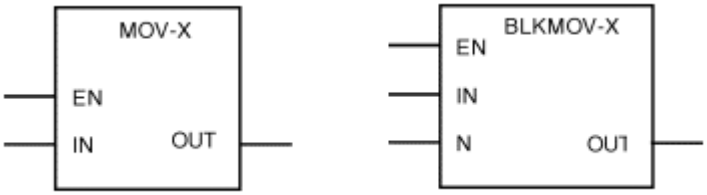

```
Network 1
LD    I0.0
+R    AC1, AC0
*R    AC1, VD100
/R    VD10, VD200
```

Add				Multiply				Divide			
4000.0	+ 6000.0	= 10000.0		400.0	* 200.0	= 80000.0		4000.0	/ 41.0	= 97.5609	
AC1	AC0	AC0		AC1	VD100	VD100		VD200	VD10	VD200	

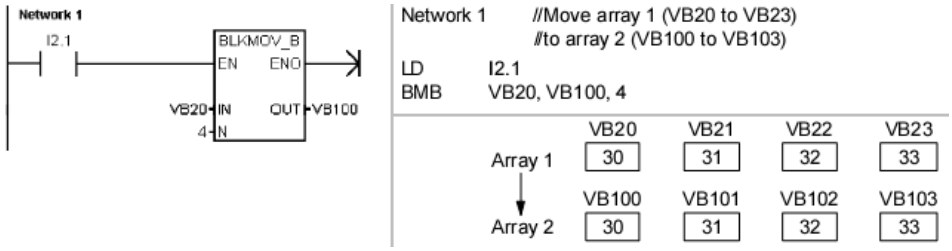


9.19 LỆNH DI CHUYỂN

Di chuyển nội dung ô nhớ hay hằng số IN ra ô nhớ OUT bằng lệnh MOV_X (STL: MOVX) hay di chuyển một khối N ô nhớ địa chỉ bắt đầu IN sang khối ô nhớ địa chỉ bắt đầu OUT dùng lệnh BLKMOV_X (STL: BMX), ô nhớ có thể là byte, word, double word hay real, X = B, W, DW, R

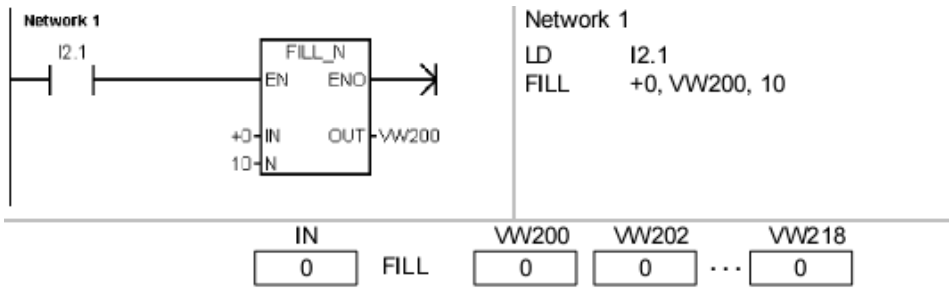


Ví dụ:

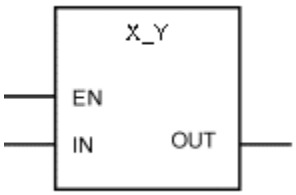


SWAP tráo đổi hai byte của một từ

FILL_N chép một từ IN vào một loạt N ô nhớ địa chỉ bắt đầu OUT



9.20 LỆNH ĐỔI



Chuyển đổi dữ liệu IN loại X sang dữ liệu OUT loại Y
B-I, I_DI, DI_R
DI_I, I_B
BCD_I, I_BCD
Các ô nhớ IN và OUT có kích thước phù hợp lệnh

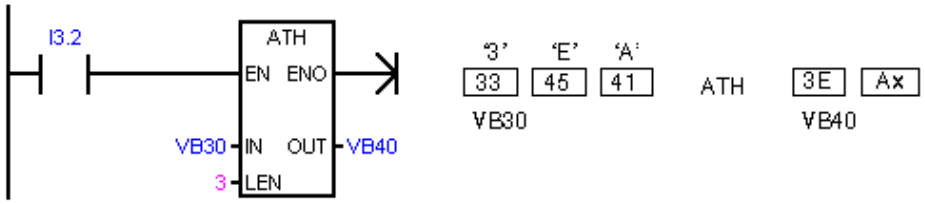
- ROUND đổi số thực sang số nguyên kép làm tròn
- TRUNC đổi số thực sang số nguyên kép bỏ phần lẻ

Ví dụ:

```
Network 1 //Convert inches to centimeters:
//1. Load a counter value (inches) into AC1.
//2. Convert the value to a real number.
//3. Multiply by 2.54 (convert to centimeters).
//4. Convert the value back to an integer.
LD I0.0
ITD C10, AC1
DTR AC1, VD0
MOVR VD0, VD8
*R VD4, VD8
ROUND VD8, VD12
Network 2 //Convert a BCD value to an integer
LD I0.3
BCDI AC0
```

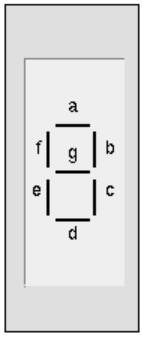
- ATH đổi một số byte ASCII (30..39, 41..46) chiều dài LEN ra các số HEX 0..9, A..F

- HTA đổi một số digit 0..F chiều dài LEN ra mã ASCII

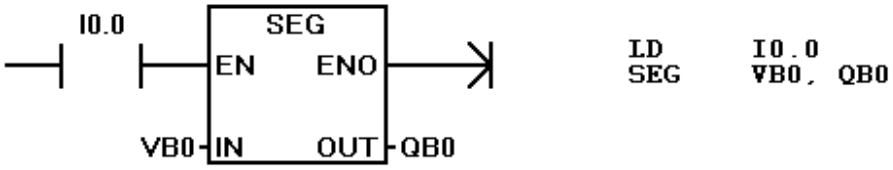


- SEG đổi số nhị phân 0-15 trong 4 bit thấp của byte in thành mã 7 đoạn trong byte out

(IN) LSD	Segment Display	(OUT) - g f e d c b a
0	0	0 0 1 1 1 1 1 1
1	1	0 0 0 0 0 1 1 0
2	2	0 1 0 1 1 0 1 1
3	3	0 1 0 0 1 1 1 1
4	4	0 1 1 0 0 1 1 0
5	5	0 1 1 0 1 1 0 1
6	6	0 1 1 1 1 1 0 1
7	7	0 0 0 0 0 1 1 1



(IN) LSD	Segment Display	(OUT) - g f e d c b a
8	8	0 1 1 1 1 1 1 1
9	9	0 1 1 0 0 1 1 1
A	A	0 1 1 1 0 1 1 1
B	b	0 1 1 1 1 1 0 0
C	C	0 0 1 1 1 0 0 1
D	d	0 1 0 1 1 1 1 0
E	E	0 1 1 1 1 0 0 1
F	F	0 1 1 1 0 0 0 1



Ví du : Chương trình đèn giao thông hiển thị giây lùi, đèn xanh Q0.0, đèn đỏ Q0.1, đèn bảy đoạn QB3, QB2

```
NETWORK1
LD    I0.0
LPS
AN    T38
TON   T37, 200
LRD
A     T37
TON   T38, 100
LRD
AN    T37
=     Q0.0
LPP
A     T37
=     Q0.1
NETWORK 2
LD    Q0.0
LPS
MOVW  +200, MW0
AENO
-I    T37, MW0
AENO
/I    +10, MW0
LRD
MOVW  MW0, MW2
```

```
IBCD   MW2
LRD
SEG    MB3, QB2
LPP
MOVB   MB3, MB4
AENO
SRB    MB4, 4
AENO
SEG    MB4, QB3
NETWORK 3
LD     Q0.1
LPS
MOVW   +100, MW10
AENO
-I     T38, MW10
AENO
/I     +10, MW10
LRD
MOVW   MW10, MW12
IBCD   MW12
LRD
SEG    MB13, QB2
LPP
MOVB   MB13, MB14
AENO
SRB    MB14, 4
AENO
SEG    MB14, QB3
```

9.21 LỆNH GHI DỜI

- SHRB dời một bit DATA vào thanh ghi chiều dài N có địa chỉ bit LSB là S_BIT. Khi N dương DATA vào LSB còn MSB dời ra SM1.1. Khi N âm DATA vào MSB còn LSB dời ra SM1.1
- SHR_X/ SHL_X dời phải/ trái một byte, từ hay từ kép IN tùy theo X = B, W, DW vào OUT. Số bit dời qui định bởi N, bit được dời ra chứa vào SM1.1 và OUT.

9.22 LỆNH QUAY

ROR_X/ ROL_X quay phải/trái một byte, từ hay từ kép IN, bit được dời ra đưa trở lại vào IN, vào SM1.1 và vào OUT. Số lần quay định bởi N. X= B, W, DW

9.23 LỆNH LOGIC

Thực hiện phép tính lodic giữa hai byte, word, double word

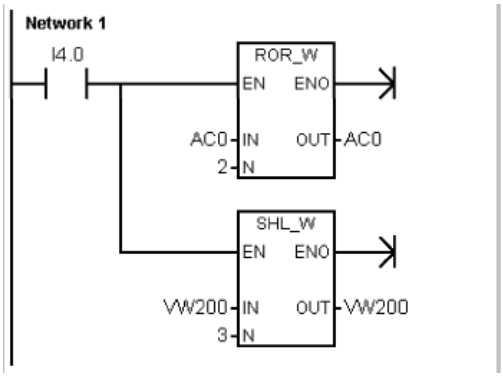
Lệnh AND : ANDB, ANDW, ANDD

Lệnh OR: ORB, ORW, ORD

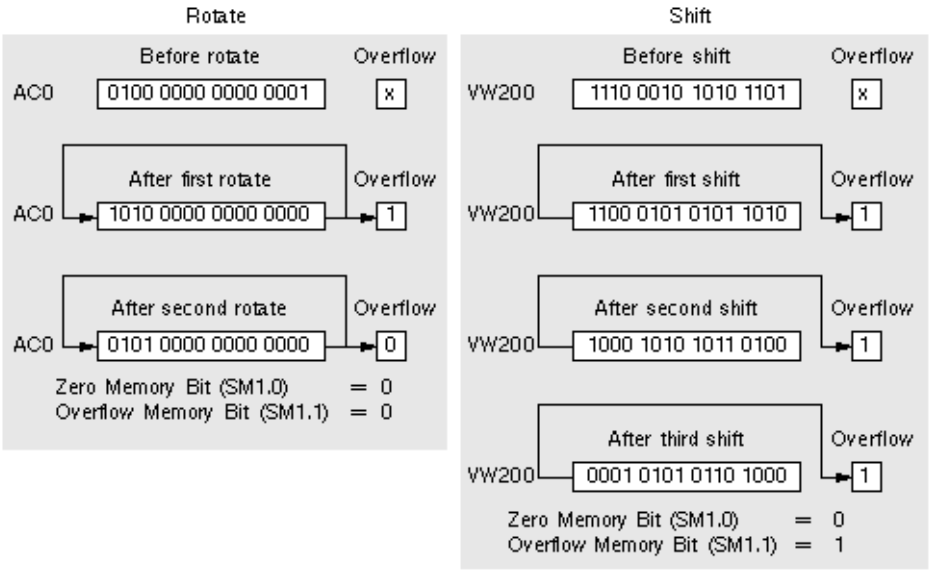
Lệnh XOR: XORB, XORW, XORD

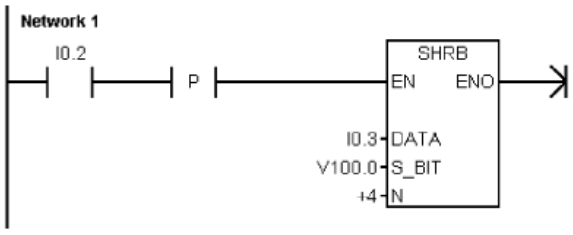
Lệnh INV: INB, INW, IND

Ví dụ:

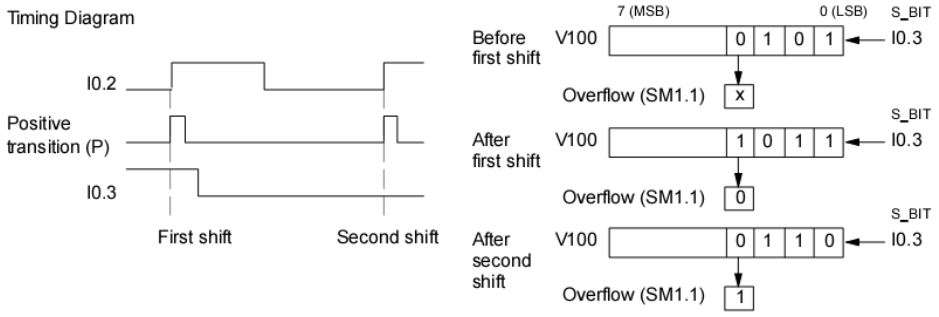


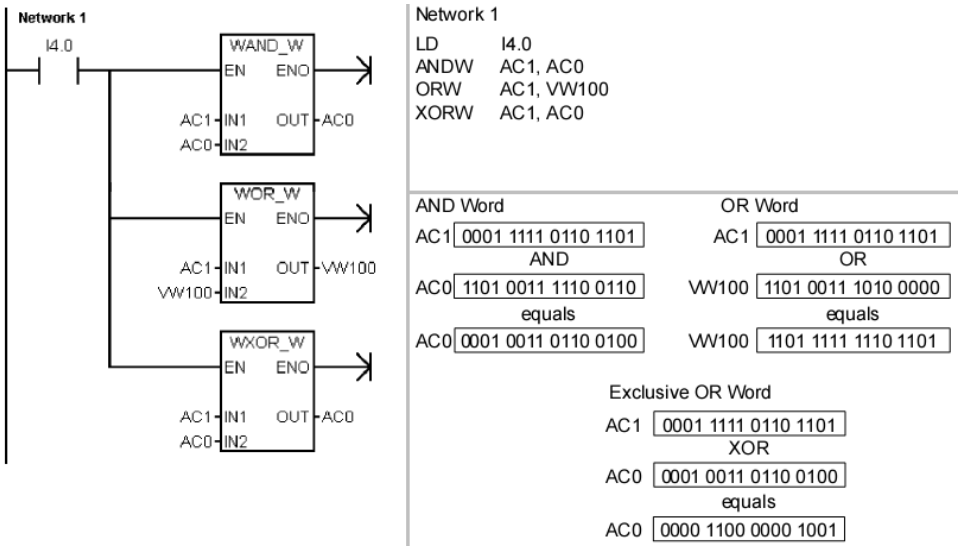

```
Network 1
LD      I4.0
RRW     AC0, 2
SLW     VW200, 3
```

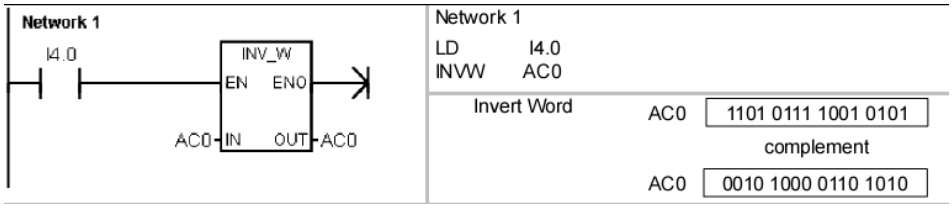





```
Network 1
LD      I0.2
EU
SHRB    I0.3, V100.0, +4
```





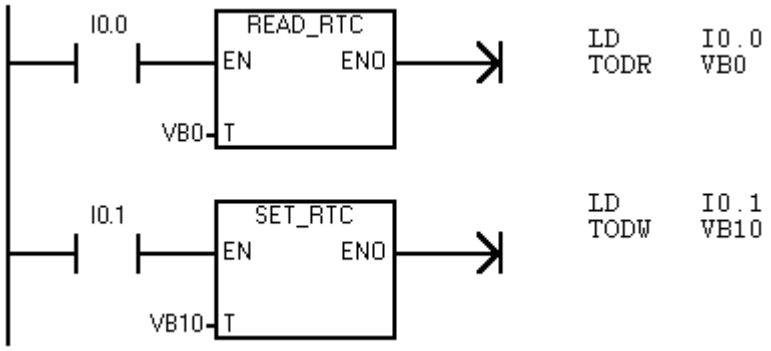


9.24 LỆNH ĐỒNG HỒ

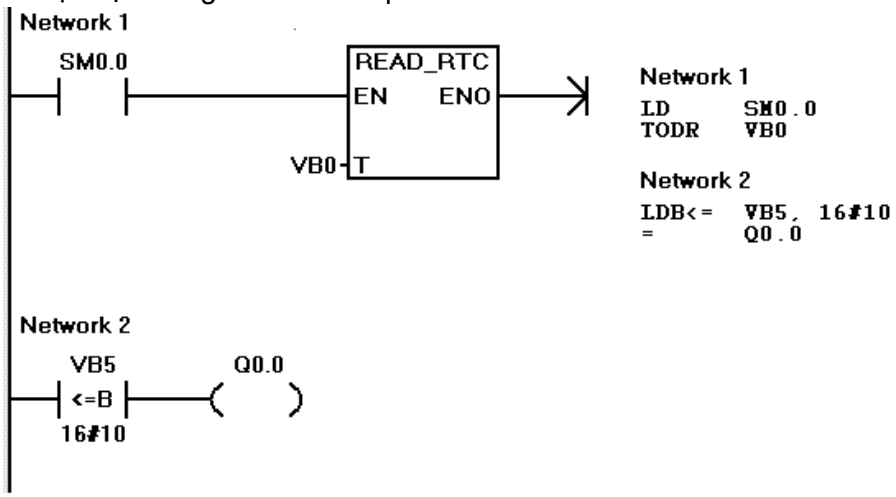
Đồng hồ trong PLC đếm thời gian ngày giờ phút giây, đọc thời gian bằng lệnh READ_RTC, chỉnh sửa thời gian bằng lệnh SET_RTC. Thời gian chứa trong bảng T 8 byte BCD có dạng sau

T	Năm 00..99
T+1	Tháng 01..12

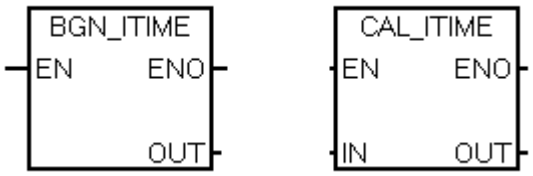
T+2	Ngày 01..31
T+3	Giờ 00..23
T+4	Phút 00..59
T+5	Giây 00..59
T+6	0
T+7	Ngày trong tuần 0..7, 1: Chủ nhật, 2: thứ hai 0: Không dùng



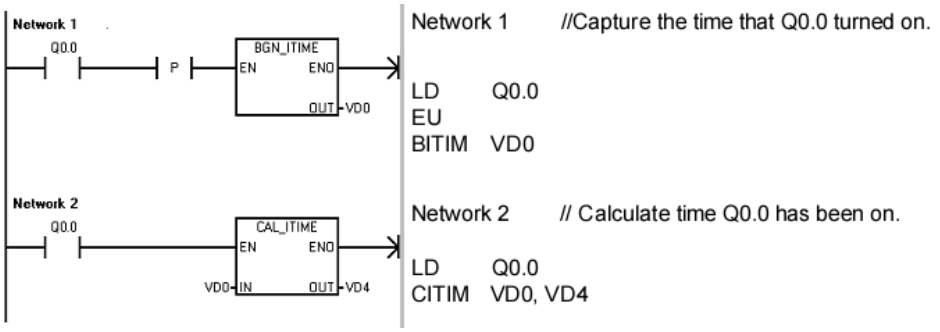
Ví dụ: Tạo xung 10 sec mỗi phút



9.25 LỆNH TÍNH KHOẢNG THỜI GIAN

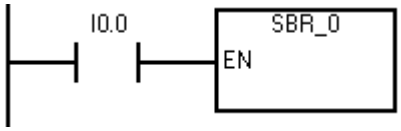


PLC có bộ đếm 32 bit đếm xung 1msec, khi EN ON lệnh BGN_ITIME (beginning interval time) đưa giá trị trong bộ đếm này vào từ kép OUT , lệnh CAL_ITIME (calculating interval time) tính hiệu số giá trị trong bộ đếm và giá trị trong IN, chứa vào OUT

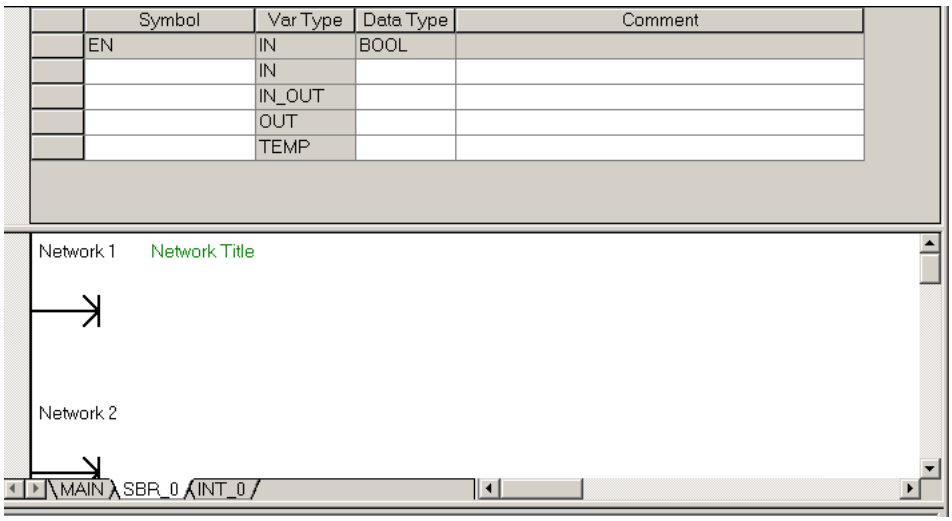


9.26 GỌI CHƯƠNG TRÌNH CON

Có thể có tối đa 64 chương trình con, chương trình con viết sau chương trình chính và được đóng khung bằng SBR_N RET. Chương trình con được gọi bằng lệnh CALL SBR_N. Chương trình con có thể kèm tham số vào ra.

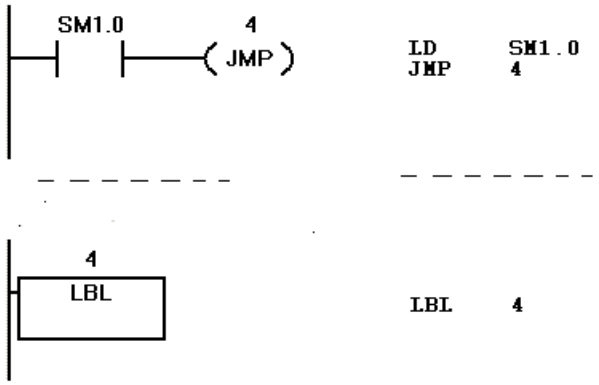


Tham số vào ra được khai báo trong bảng biến cục bộ của chương trình con

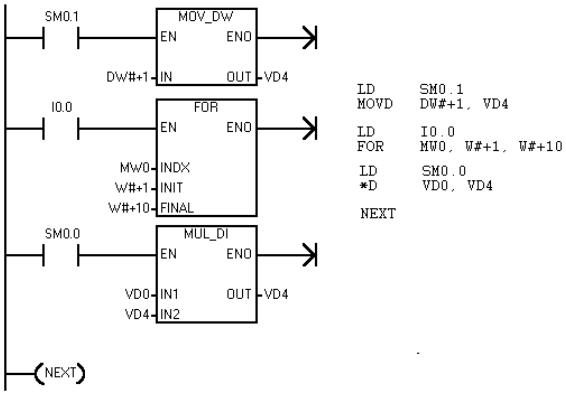


9.27 LỆNH ĐIỀU KHIỂN CHƯƠNG TRÌNH

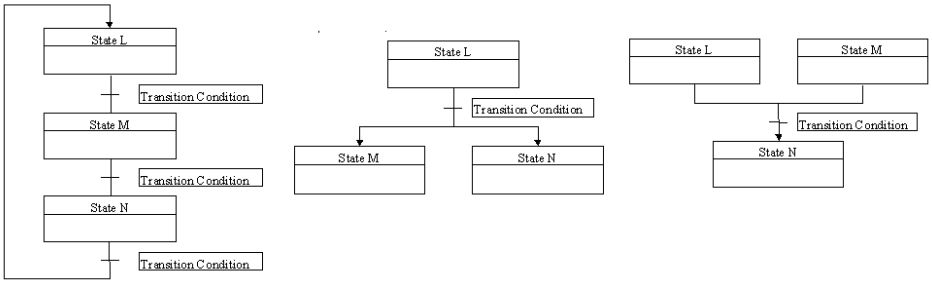
Lệnh JMP n nhảy đến đoạn chương trình nhãn n



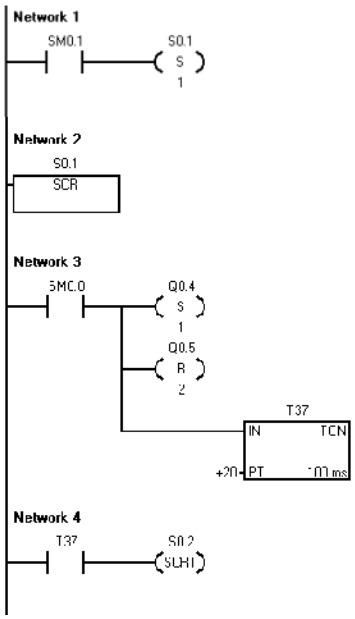
Lệnh FOR NEXT thực hiện lập vòng các dòng lệnh giữa FOR và NEXT, số vòng được đếm bởi biến INDX có giá trị đầu INIT và giá trị cuối FINAL, số vòng lặp là FINAL -INIT +1. Có thể đặt các vòng lặp lồng vào nhau.



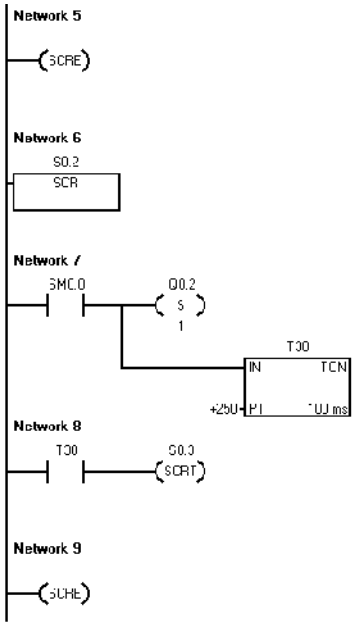
Lệnh SCR Sequence Control Relay dùng để thực hiện một đoạn chương trình lặp, đoạn chương trình này đóng khung bởi lệnh SCR và SCRE, được thực hiện lặp khi bit điều khiển tương ứng vùng nhớ Sx.y là ON ứng với một trạng thái. Muốn chuyển sang trạng thái khác ta dùng lệnh SCRT cho một bit S khác ON và tắt bit S của trạng thái hiện tại. Có ba loại chuyển tiếp trạng thái như ở hình dưới đây



Ví dụ:

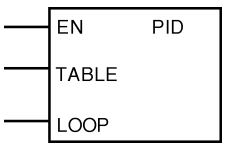



```
Network 1       //On the first scan enable State 1.
LD      SM0.1
S       S0.1, 1

Network 2       //Beginning of State 1 control region.
LSCR    S0.1

Network 3       //Control the signals for Street 1:
                //1. Set: Turn on the red light.
                //2. Reset: Turn off the yellow and green lights.
                //3. Start a 2-second timer.
LD      SM0.0
S       Q0.4, 1
R       Q0.5, 2
TON     T37, +20

Network 4       //After a 2 second delay, transition to State 2.
LD      T37
SCRT    S0.2

Network 5       //End of SCR region for State 1.
SCRE
```




```
Network 6       //Beginning of State 2 control region.
LSCR    S0.2

Network 7       //Control the signals for Street 2:
                //1. Set: Turn on the green light.
                //2. Start a 25-second timer.
LD      SM0.0
S       Q0.2, 1
TON     T38, +250

Network 8       //After a 25 second delay, transition to State 3.
LD      T38
SCRT    S0.3

Network 9       //End of SCR region for State 2.
SCRE
```

9.32 LỆNH PID



Lệnh này không có ở CPU 212, 214, trong chương trình có thể dùng tối đa 8 lệnh PID. Thông số lệnh chứa trong một bảng 80 byte gồm 9 thông số thực 4 byte, các byte còn lại

dùng cho Auto tuning. TABLE là địa chỉ bảng (vùng VB), LOOP từ 0 đến 7. Sau đây là bảng các thông số

Bảng 9.4

Địa chỉ lệnh	Tên	Miêu tả
0	Đại lượng đo hiện tại PV_n	Đại lượng được điều khiển, chuẩn hóa từ 0.0 đến 1.0
4	Đại lượng đặt SV_n	Trị đặt, chuẩn hóa
8	Đại lượng điều khiển M_n	Ngõ ra của PID, chuẩn hóa
12	Độ lợi vòng K_C	Hệ số tỷ lệ
16	Thời gian lấy mẫu T_S	Thời gian lấy mẫu, đơn vị là giây
20	Thời gian tích phân T_I	Đơn vị phút
24	Thời gian vi phân T_D	Đơn vị phút
28	Đại lượng offset MI_{n-1}	Giá trị tích phân ở thời điểm trước, chuẩn hóa
32	Đại lượng đo trước PV_{n-1}	Trị số đo của đại lượng được điều khiển ở thời gian lấy mẫu trước

Thuật toán PID được thực hiện theo phương trình:

$$M_n = MP_n + MI_n + MD_n$$

$$MP_n = K_c^*(SV_n - PV_n)$$

$$MI_n = MI_{n-1} + K_c^* T_S / T_I^*(SV_n - PV_n)$$

$$MD_n = K_c^* T_D / T_S^*(PV_{n-1} - PV_n)$$

Trị đặt SV và trị đo PV đều phải là số thực và được chuẩn hóa. Đầu tiên phải đổi số nguyên 16 bit ra số thực, sau đó chia cho tầm (Span) của đại lượng đó và cộng với Offset (là 0.0 nếu đơn cực và 0.5 nếu lưỡng cực)

Ví dụ, ta muốn đổi trị số đo của AIW0 ra số thực và chuẩn hóa cho 64000 cất vào VD100:

```
XORD    AC0, AC0            //Xóa ACC
MOVW    AIW0, AC0           //Cất trị đo vào ACC
LDW>=   AC0, 0              //Nếu dương
JMP     0                   //đổi sang số thực
NOT                         //Nếu âm
ORD     16#FFFF0000, AC0    //Khai triển dấu giá trị trong AC0
LBL     0
```

```
DTR     AC0, AC0        //Đổi số nguyên 32 bit ra số thực
/R      64000.0, AC0    //Chuẩn hóa
+R      0.5, AC0
MOVR    AC0, VD100
```

Tín hiệu ra M là đại lượng đã chuẩn hóa, muốn dùng để điều khiển ta phải đổi ra số nguyên 16 bit.

Ví du: đổi trị thực lưỡng cực ở VD108 ra số nguyên 16 bit ở AQW0

```
MOVR    VD108, ACV0
-R      0.5, AC0
*R      64000.0, AC0
TRUNC   AC0, AC0
MOVW    AC0, AQW0
```

Ví dụ: lập trình PID điều khiển mức nước trong bồn chứa sao cho áp suất nước ở đường xả không đổi, mức nước được duy trì ở mức 75% tối đa bằng cách điều khiển vận tốc bơm.

Đo mức: AIW0 Điều khiển bơm: AQW0

Bảng PID: VD100 Gọi PID: gọi ngắt thời gian 0.1s

K_C : 0.25, T_S : 0.1s, T_I: 30 phút

Chương trình dạng STL:

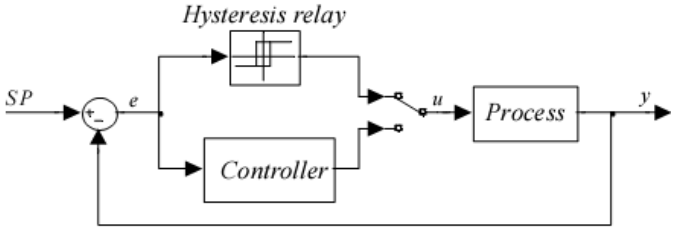
```
//Main
Network 1
LD      SM0.1
MOVR    0.75, VD104     // Điểm đặt 75%
MOVR    0.25, VD112     // Độ lợi vòng
MOVR    0.10, VD116     // TS
MOVR    30.0, VD120     // TI
MOVR    0.0, VD124      // Không dùng đạo hàm
MOVB    100, SMB34      // Ngắt thời gian 0.1s gọi PID
ATCH    0,10
ENI
//INT0
Network 1
LD      SM0.0
XORD    AC0, AC0
MOVW    AIW0, AC0       // Đọc mức nước và đổi sang số thực rồi
                        đưa vào bảng
DTR     AC0, AC0
/ R     32000.0, AC0
MOVR    AC0, VD100
Network 2
LD      I0.0            // Khi I0.0 ON thì điều khiển PID
PID     VB100, 0
```

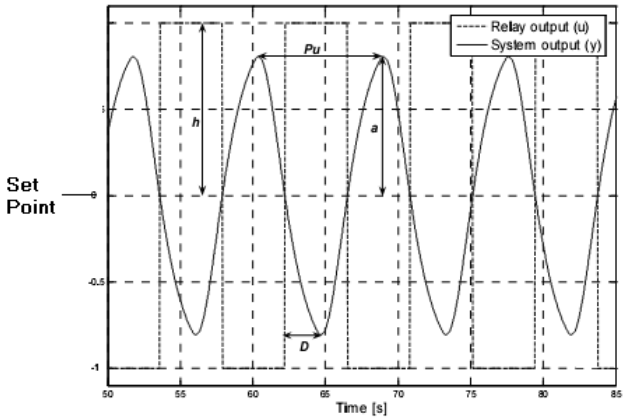
```
Network 3
LD          SM0.0
MOVR        VD108, AC0      // Xuất ra điều khiển
*R          32000.0, AC0
TRUNC       AC0, AC0
MOVW        AC0, AQW0
```

Step7 Microwin có PID wizard giúp lập trình PID, tạo ra chương trình con PID_INIT và chương trình ngắt PID_EXE, chương trình chính gọi PID_INIT mỗi chu kỳ quét, còn PID_INIT gọi PID_EXE

```
LD    SM0.0
CALL  PID0_INIT, AIW0, 0.75, AQW0
```

Auto tune control panel giúp chọn các thông số K_C, T_I, T_D, dùng thuật toán của K. J.Astrom and T. Hagglund. PLC cần kết nối với máy tính và quá trình để thực hiện tuning.Sử dụng điều khiển on/off để biến ngõ ra dao động quanh điểm đặt, từ đó suy ra Chu kỳ giới hạn w_u và độ lợi giới hạn K_u



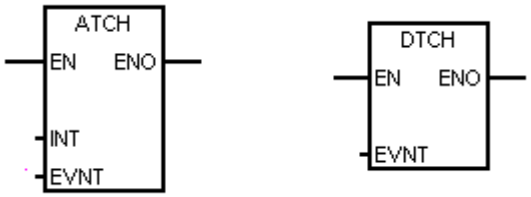


$$\omega_u = \frac{2\pi}{P_u}$$

$$K_u = \frac{4h}{\pi a}$$

9.28 LỆNH NGẮT

Có thể có tối đa 128 chương trình phục vụ ngắt, viết sau chương trình con, đóng khung bằng INT_n RETI. Cấm/ cho phép ngắt bằng lệnh ENI/DISI. INT_n được gọi đến khi xảy ra sự kiện EVNT. Có tất cả 27 sự kiện có thể gây ra ngắt. Sự kiện được liên kết với INT_n thông qua lệnh gắn ATCH và tháo DTCH.



Bảng 9.5: *Các sự kiện ngắt và ưu tiên*

Số Sự kiện	Miêu tả	Ưu tiên	Ưu tiên trong nhóm
8	Nhận ký tự ở Port 0	Cao nhất	1
9	Truyền xong Port 0		1
23	Nhận xong bản tin ở Port 0		1
24	Nhận xong bản tin ở Port 1		2
25	Nhận ký tự ở Port 1		2
26	Truyền xong Port 1		2
0	Ngắt ở cạnh lên của I0.0	Giữa	1
2	Ngắt ở cạnh lên của I0.1		2
4	Ngắt ở cạnh lên của I0.2		3
6	Ngắt ở cạnh lên của I0.3		4
1	Ngắt ở cạnh xuống của I0.0		5
3	Ngắt ở cạnh xuống của I0.1		6
5	Ngắt ở cạnh xuống của I0.2		7
7	Ngắt ở cạnh xuống của I0.3		8
12	Đếm vận tốc cao HSC0: trị đo bằng trị đặt		1
13	Đếm vận tốc cao HSC1: trị đo bằng trị đặt		9
14	HSC1 đổi hướng đếm		10
15	Xóa ngoài HSC1		11
16	Đếm vận tốc cao HSC2: trị đo bằng trị đặt		12

17	HSC2 đổi hướng đếm		13
18	Xóa ngoài HSC2		14
32	HSC3: trị đo bằng trị đặt		19
29	HSC4: trị đo bằng trị đặt		20
30	HSC4 đổi hướng đếm		21
31	Xóa ngoài HSC4		22
33	HSC5: trị đo bằng trị đặt		23
19	Đếm xung PLS0 xong		15
20	Đếm xung PLS1 xong		16
10	Ngắt thời gian 0	Thấp nhất	1
11	Ngắt thời gian 1		2
21	Ngắt timer T32		3
22	Ngắt timer T96		4

Tùy theo loại CPU có một số sự kiện không được hỗ trợ. Sau đây ta sẽ phân tích một số ngắt chính.

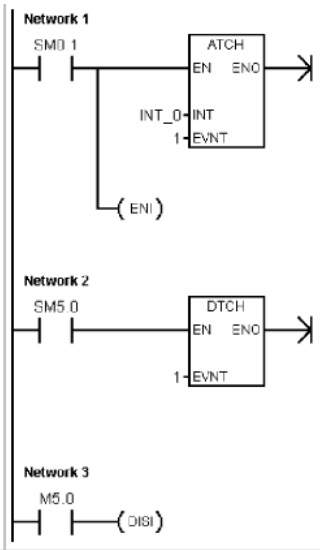
9.28.1 Ngắt thời gian

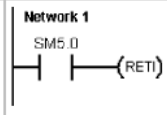
Ngắt thời gian 0/1 xảy ra theo chu kỳ ấn định (tối đa 255ms) bởi nội dung của SMB34/SMB35 (đơn vị ms) thường dùng để đọc hay xuất tín hiệu analog. Ngắt timer T32/T96 xảy ra khi timer T32 hoặc T96 hoàn tất thời gian trễ đã đặt.

Ví dụ:

```
NETWORK 1 // Subroutine 0
LD SM0.0
MOVB 100, SMB34 // Chu kỳ ngắt 100ms
ATCH INT_0 10 // Liên kết sự kiện số 10 với  INT_0
ENI // Cho phép ngắt toàn cục
NETWORK 1 // Interrupt 0
// Đọc giá trị AIW4 mỗi 100ms
LD SM0.0
MOVW AIW4, VW100
```

9.28.2 Ngắt ngõ vào

MAIN		Network 1 //On the first scan:
		//1. Define interrupt routine INT_0 to
		// be a falling-edge interrupt for I0.0
		//2. Globally enable interrupts.
		LD SM0.1
		ATCH INT_0, 1
		ENI
		Network 2 //If an I/O error is detected,
		//disable the falling-edge interrupt for I0.0.
		//This network is optional.
		LD SM5.0
		DTCH 1
		Network 3 //When M5.0 is on,
		//disable all interrupts.
		LD M5.0
		DISI

INT 0		Network 1 //I0.0 falling-edge interrupt routine: //Conditional return based on an I/O error. LD SM5.0 CRETI

9.30 Tạo xung tần số cao và điều rộng xung

Lệnh PLS phát dãy xung vuông tần số thay đổi, số xung thay đổi PTO hay dãy xung điều rộng PWM có chu kỳ thay đổi ở ngõ ra Q0.0 hay Q0.1. Dùng SMB67/SMB77 cài đặt cấu hình phát xung

Q0.0	Q0.1		Control Bits	
SM67.0	SM77.0	PTO/PWM update cycle time value	0 = no update;	1 = update cycle time
SM67.1	SM77.1	PWM update pulse width time value	0 = no update;	1 = update pulse width
SM67.2	SM77.2	PTO update pulse count value	0 = no update;	1 = update pulse count
SM67.3	SM77.3	PTO/PWM time base select	0 = 1 µs/tick;	1 = 1ms/tick
SM67.4	SM77.4	PWM update method:	0 = asynchronous update,	1 = synchronous update
SM67.5	SM77.5	PTO operation:	0 = single segment operation	1 = multiple segment operation
SM67.6	SM77.6	PTO/PWM mode select	0 = selects PTO;	1 = selects PWM
SM67.7	SM77.7	PTO/PWM enable	0 = disables PTO/PWM;	1 = enables PTO/PWM

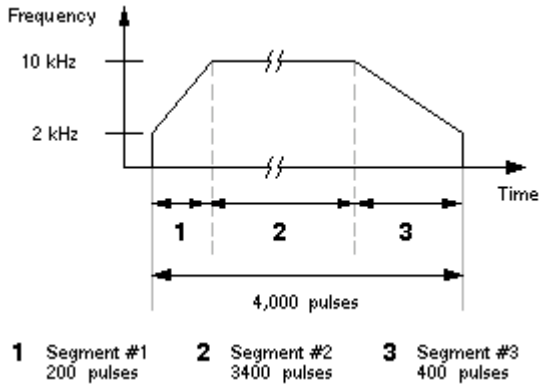
Dùng các ô nhớ 16 bit chứa chu kỳ và bề rộng xung, ô nhớ 32 bit chứa số lượng xung. Trước khi dùng lệnh PLS phải xóa Q0.0 và Q0.1, lệnh PLS còn được dùng mỗi khi thay đổi thông số dãy xung.

Q0.0	Q0.1	Other PTO/PWM Registers
SMW68	SMW78	PTO/PWM cycle time value (range: 2 to 65535)
SMW70	SMW80	PWM pulse width value (range: 0 to 65535)
SMD72	SMD82	PTO pulse count value (range: 1 to 4294967295)
SMB166	SMB176	Number of segment in progress (used only in multiple segment PTO operation)
SMW168	SMW178	Starting location of profile table, expressed as a byte offset from V0 (used only in multiple segment PTO operation)
SMB170	SMB180	Linear profile status byte
SMB171	SMB181	Linear profile result register
SMB172	SMB182	Manual mode frequency register

Mode PWM tạo xung có chu kỳ từ 10 microsec đến 65535 microsec, 2 ms đến 65535 ms , bề rộng xung từ 0 microsec (msec) đến 65535 microsec (msec), Sự thay đổi đặc tính xung thực hiện đồng bộ ở cuối mỗi chu kỳ xung, trong trường hợp thay đổi cả đơn vị thời gian thì thực hiện thay đổi ngay tức khắc gọi là cập nhật bất đồng bộ

Mode PTO (pulse train output) phát xung vuông có chu kỳ thay đổi 10 microsec đến 65535 microsec, 2 ms đến 65535 ms , số lượng xung từ 1 đến 4.294.967.295. Trong suốt dãy xung, chu kỳ có thể thay đổi hay giữ nguyên. Ở chế độ single segment pipelining, giả sử chia thời gian thành nhiều khoảng, ta qui định số lượng xung và chu kỳ, chứa vào ô nhớ SMW68/78 và SMD72/82, sau đó thực hiện lênh PLS; sử dụng ngắt khi đã phát đủ số lượng xung để thực hiện lệnh PLS với số lượng xung và chu kỳ mới.

Trường hợp multisegment pipelining, ta lập trình trước biểu đồ thay đổi trong một bảng gọi là profile table. Ví dụ sau điều khiển động cơ bước với biểu đồ vận tốc gồm ba đoạn tăng tốc, vận tốc không đổi và giảm tốc



Tần số xung ban đầu là 2KHz , tần số cao nhất là 10 KHz tương ứng chu kỳ là 500ms và 100ms. Trong giai đoạn tăng tốc phải phát 200 xung với chu kỳ giảm từ 500ms xuống 100ms, còn

giai đoạn giảm tốc, chu kỳ tăng từ 100ms lên 500ms và phải phát 400 xung, vậy ta phải xác định

gia số chu kỳ = (chu kỳ cuối –chu kỳ đầu)/số xung là số nguyên

đoạn tăng tốc gsck là -2, đoạn giữa, là 0 và đoạn cuối , là 1

Các thông số của mỗi đoạn chứa trong một bảng ô nhớ, giả sử địa chỉ ban đầu là VB100

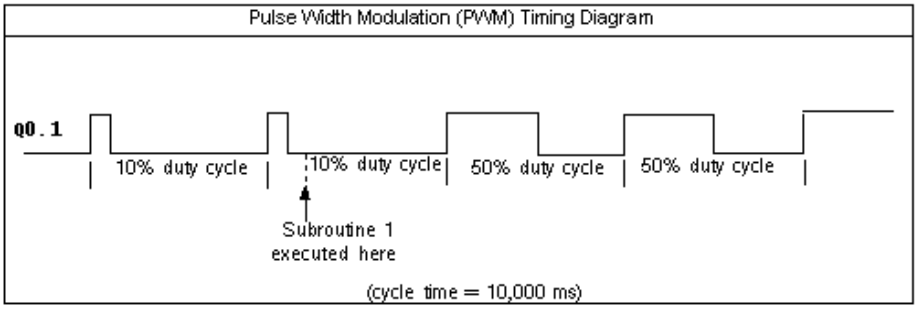
VB100	3	Số đoạn	
VW101	500	Chu kỳ đầu	Đoạn 1
VW103	-2	Gia số chu kỳ	
VD105	200	Số xung	
VW109	100	Chu kỳ đầu	Đoạn 2
VW111	0	Gia số chu kỳ	
VD113	3400	Số xung	
VW117	100	Chu kỳ đầu	Đoạn 3
VW119	1	Gia số chu kỳ	
VD121	400	Số xung	

Để điều khiển trơn, chu kỳ cuối đoạn trước nên bằng chu kỳ đầu đoạn sau

Các bit sau cho biết trạng thái của PTO

Q0.0	Q0.1	Status Bits	
SM66.4	SM76.4	PTO profile aborted due to delta calculation error	
		0 = no error;	1 = aborted
SM66.5	SM76.5	PTO profile aborted due to user command	
		0 = no abort;	1 = aborted
SM66.6	SM76.6	PTO pipeline overflow/underflow	
		0 = no overflow; 1 = overflow/underflow	
SM66.7	SM76.7	PTO idle	0 = in progress; 1 = PTO idle

Ví dụ: tạo xung PWM



// Main, Reset Q0.1 , gọi SBR_0

NETWORK 1

LD SM0.1

R Q0.1 1

CALL SBR_0

NETWORK 2 // Khi M0.0 On thay đổi bề rộng xung 50%

LD M0.0

EU

CALL SBR_1

//SBR0

NETWORK 1

LD SM0.0

MOVB 16#DB, SMB77 // Byte điều khiển, PWM, đơn vị ms, cập nhật đồng bộ

MOVW +10000, SMW78 // Chu kỳ 10,000 ms

MOVW +1000, SMW80 // Bề rộng xung 1,000 ms

PLS 1 // Phát xung ở Q0.1

MOVB 16#DA, SMB77 // Nạp byte điều khiển cho lần sau

//SBR 1

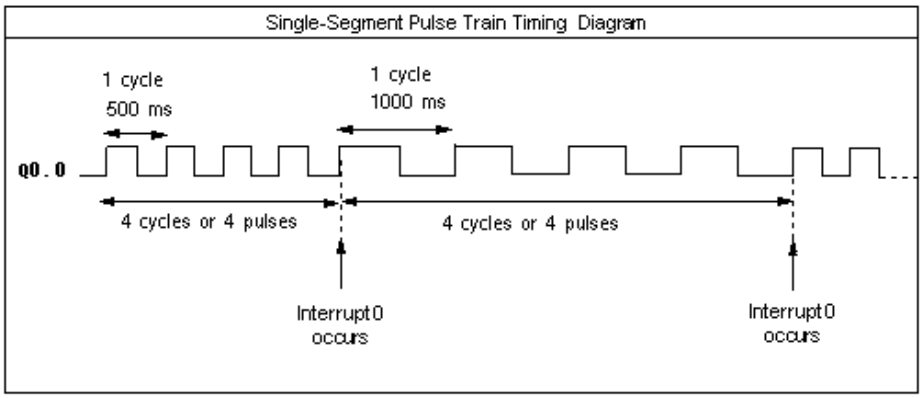
NETWORK 1

LD SM0.0

MOVW +5000, SMW80 // Bề rộng xung 5000 ms

PLS 1 // Phát xung

Ví dụ: tạo xung PTO một doạn



// Main, phát dãy 4 xung PTO chu kỳ thay phiên 500ms rồi 1000ms

NETWORK 1

LD SM0.1

R Q0.0 1

CALL SBR_0

// SBR 0

NETWORK 1

LD SM0.0

MOVB 16#8D, SMB67 // Byte điều khiển, PTO, một đoạn, đơn vị ms:

MOVW +500, SMW68 // Chu kỳ 500ms.

MOVD +4, SMD72 // Phát 4 xung.

ATCH INT_0, 19 // Gọi ngắt 19 khi phát xong

ENI

PLS 0 // phát xung ở Q0.0

MOVB 16#89, SMB67 // Nạp byte điều khiển cho đoạn sau

// Interrupt 0

NETWORK 1

// Nếu chu kỳ là 500 ms: đổi chu kỳ là 1000 ms và phát 4 xung

LDW= SMW68 +500

MOVW +1000 SMW68

PLS 0

CRETI

NETWORK 2

// Nếu chu kỳ là 1000 ms: đổi chu kỳ là 500 ms và phát 4 xung

LDW= SMW68 +1000

MOVW +500 SMW68

PLS 0

Ví dụ: phát xung PTO nhiều đoạn

```
// Main
NETWORK 1
LD SM0.1
R Q0.0 1
CALL SBR_0
// Subroutine 0
NETWORK 1
 // Nạp bản profile
LD SM0.0
MOVB 3 VB500 // Số đoạn là 3
 // Segment 1:
MOVW +500 VW501 // Chu kỳ ban đầu 500 ms
MOVW -2 VW503 // Gia số chu kỳ  -2 ms
MOVD +200 VD505 // Số xung  200
 // Segment 2:
MOVW +100 VW509 //Chu kỳ đầu 100 ms
MOVW +0 VW511 // Gia số 0 ms
MOVD +3400 VD513 // Số đoạn  3400
 // Segment 3:
MOVW +100 VW517 // Chu kỳ đầu  100 ms
MOVW +1 VW519 // Gia số  1 ms
MOVD +400 VD521 // Số xung là 400
NETWORK 2
LD SM0.0
MOVB 16#A8 SMB67 // Byte điều khiển, PTO, nhiều đoạn, đơn vị ms,
MOVW +500 SMW168 // Bảng bắt đầu ở VB500.
ATCH INT_0 19 // Gọi ngắt 0 khi kết thúc
ENI
PLS  0 // Phát xung ở  Q0.0
MOVB 16#89 SMB67 // Nạp byte điều khiển cho lần sau
//INT0, khi phát đủ xung, cho Q0.5 on
NETWORK 1
LD SM0.0
= Q0.5
```

9.31 ĐẾM XUNG VẬN TỐC CAO

Bộ đếm vận tốc cao dùng để đếm xung tần số cao vài chục KHz từ encoder, CPU224, 224XP, 226 có 6 bộ đếm HSC0.. HSC5, CPU 221,222 có 4 bộ đếm HSC0, HSC 3..HSC5. Có bốn loại bộ đếm: đếm 1 pha với điều khiển hướng đếm bên trong hoặc bên

ngoài, đếm hai pha với hai xung nhịp cho hai chiều, đếm hai pha với hai xung nhịp vuông pha A/B. Bộ đếm có thể được điều khiển bằng tín hiệu Reset và Start. Khi Start tích cực, bộ đếm được cho phép, khi Start không tích cực, bộ đếm ngừng đếm và giữ nguyên giá trị. Khi Start tích cực, nếu Reset tích cực thì xóa bộ đếm. HSC thường dùng kết hợp với ngắt. Có tất cả 13 mode hoạt động cho HSC.

Mode	Description	Inputs			
	HSC0	I0.0	I0.1	I0.2	
	HSC1	I0.6	I0.7	I1.0	I1.1
	HSC2	I1.2	I1.3	I1.4	I1.5
	HSC3	I0.1			
	HSC4	I0.3	I0.4	I0.5	
	HSC5	I0.4			
0	Single-phase counter with internal direction control	Clock			
1		Clock		Reset	
2		Clock		Reset	Start
3	Single-phase counter with external direction control	Clock	Direction		
4		Clock	Direction	Reset	
5		Clock	Direction	Reset	Start
6	Two-phase counter with 2 clock inputs	Clock Up	Clock Down		
7		Clock Up	Clock Down	Reset	
8		Clock Up	Clock Down	Reset	Start
9	A/B phase quadrature counter	Clock A	Clock B		
10		Clock A	Clock B	Reset	
11		Clock A	Clock B	Reset	Start
12	Only HSC0 and HSC3 support mode12. HSC0 counts the number of pulses going out of Q0.0. HSC3 counts the number of pulses going out of Q0.1.				

Lập trình HSC theo các bước sau:

- Đặt byte điều khiển.
- Xác định mode và số HSC dùng lệnh HDEF, lệnh này chỉ dùng một lần cho mỗi HSC.
- Đặt giá trị ban đầu.
- Đặt giá trị so sánh
- Cài đặt ngắt
- Kích hoạt dùng lệnh HSC

Byte điều khiển cho phép hay cấm HSC, chọn hướng

đếm,mức tích cực của Reset và Start, nạp giá trị hiện tại và giá trị so sánh

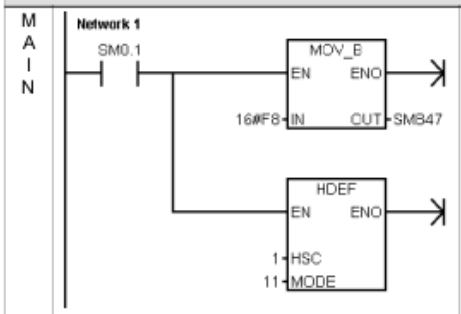
Byte điều khiển xác định trong bảng sau:

HSC0	HSC1	HSC2	HSC4	Description (used only when HDEF is executed)
SM37.0	SM47.0	SM57.0	SM147.0	Active level control bit for Reset[1]: 0 = Reset is active high 1 = Reset is active low
---	SM47.1	SM57.1	---	Active level control bit for Start[1]: 0 = Start is active high 1 = Start is active low
SM37.2	SM47.2	SM57.2	SM147.2	Counting rate selection for quadrature counters: 0 = 4X counting rate 1 = 1X counting rate

1 The default setting of the reset input and the start input are active high, and the quadrature counting rate is 4x (or four times the input clock frequency).

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	Description
SM37.3	SM47.3	SM57.3	SM137.3	SM147.3	SM157.3	Counting direction control bit: 0 = Count down 1 = Count up
SM37.4	SM47.4	SM57.4	SM137.4	SM147.4	SM157.4	Write the counting direction to the HSC: 0 = No update 1 = Update direction
SM37.5	SM47.5	SM57.5	SM137.5	SM147.5	SM157.5	Write the new preset value to the HSC: 0 = No update 1 = Update preset
SM37.6	SM47.6	SM57.6	SM137.6	SM147.6	SM157.6	Write the new current value to the HSC: 0 = No update 1 = Update current value
SM37.7	SM47.7	SM57.7	SM137.7	SM147.7	SM157.7	Enable the HSC: 0 = Disable the HSC 1 = Enable the HSC

Example: High-Speed Counter Definition Instruction




```
Network 1     //On the first scan:
              //1.  Select the start and reset
              //    inputs to be active high
              //    and select 4x mode.
              //2.  Configure HSC1 for
              //    quadrature mode with reset
              //    and start inputs
LD      SM0.1
MOVB    16#F8, SMB47
HDEF    1, 11
```

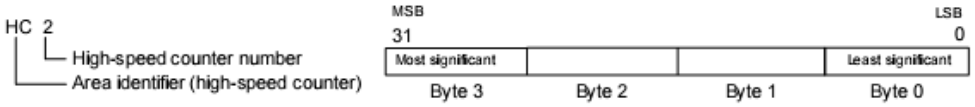
Gía trị ban đầu và giá trị so sánh của HSC là số có dấu 32 bit có địa chỉ trong bảng sau:

Value to be Loaded	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5
New current value	SMD38	SMD48	SMD58	SMD138	SMD148	SMD158
New preset value	SMD42	SMD52	SMD62	SMD142	SMD152	SMD162

Các giá trị này chỉ được nạp vào sau khi thực hiện lệnh HSC n và byte điều khiển cho phép update.

Gía trị hiện tại của HSC là ô nhớ 32 bit chỉ đọc có địa chỉ

HCn



Trạng thái của HSC được xác định bởi byte trạng thái

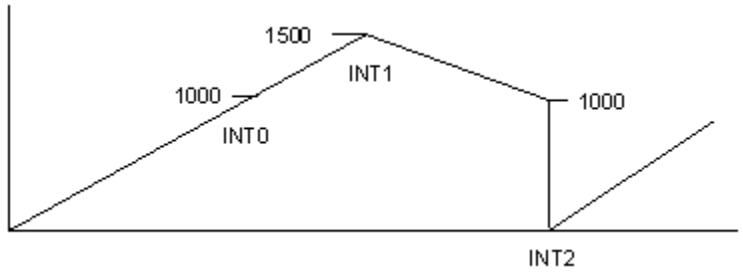
HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	Description
SM36.5	SM46.5	SM56.5	SM136.5	SM146.5	SM156.5	Current counting direction status bit: 0 = Counting down 1 = Counting up
SM36.6	SM46.6	SM56.6	SM136.6	SM146.6	SM156.6	Current value equals preset value status bit: 0 = Not equal 1 = Equal
SM36.7	SM46.7	SM56.7	SM136.7	SM146.7	SM156.7	Current value greater than preset value status bit: 0 = Less than or equal 1 = Greater than

Ví dụ:

```
//Main
NETWORK 1
LD SM0.1
MOVB 16#F8 SMB47 // Cấu hình  HSC1:
 // - Cho phép đếm
 // - Ghi giá trị mới hiện tại
 // - Ghi giá trị preset
 // - Ban đầu đếm tăng
 // - Ngõ vào start reset  tích cực cao

 // - Mode 4x
HDEF 1 11 // HSC1:quadrature mode, reset, start
MOVD +0 SMD48 // Xóa giá trị hiện tại của HSC1
MOVD +50 SMD52 // HSC1 preset value là 50
ATCH INT_0 13 // Khi HSC1 current value = preset value (EVENT 13)
 // gọi INT0
ENI
HSC 1 // Lập trình HSC1
//INT0
NETWORK 1
LD SM0.0
MOVD +0 SMD48 // Xóa giá trị hiện tại của HSC1
MOVB 16#C0 SMB47 // Ghi giá trị mới cho HSC1
```

HSC 1 // Program HSC1

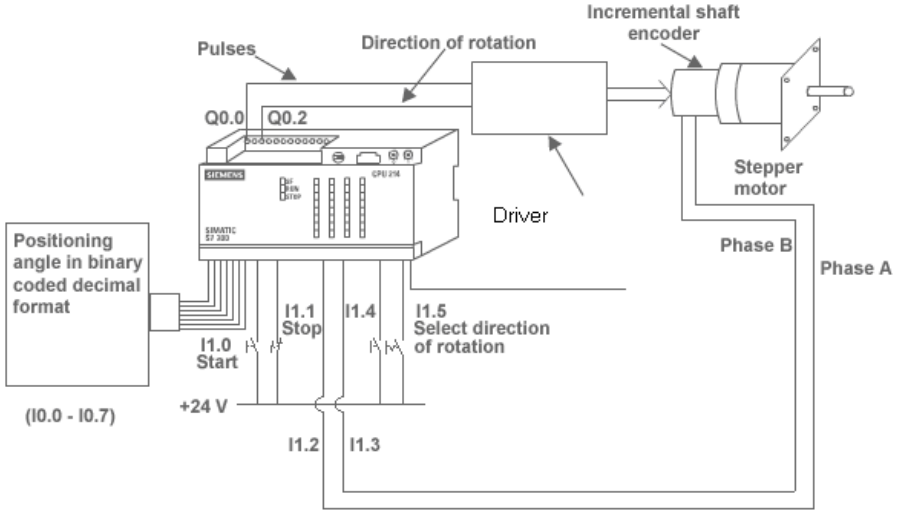
Ví dụ: điều khiển phát xung và đếm xung

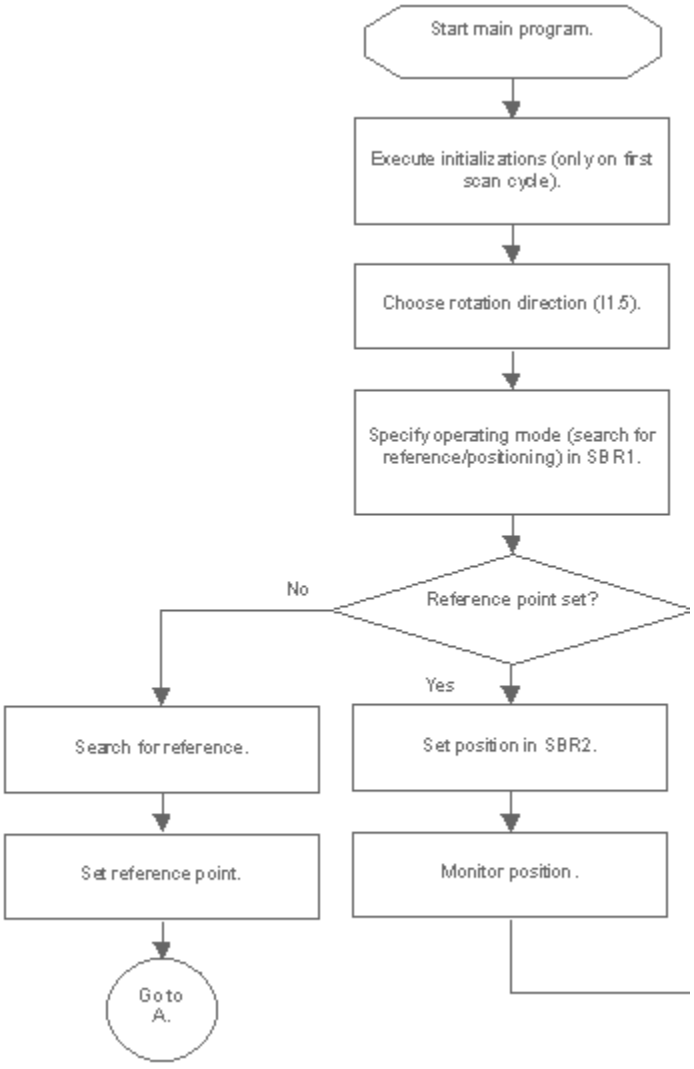


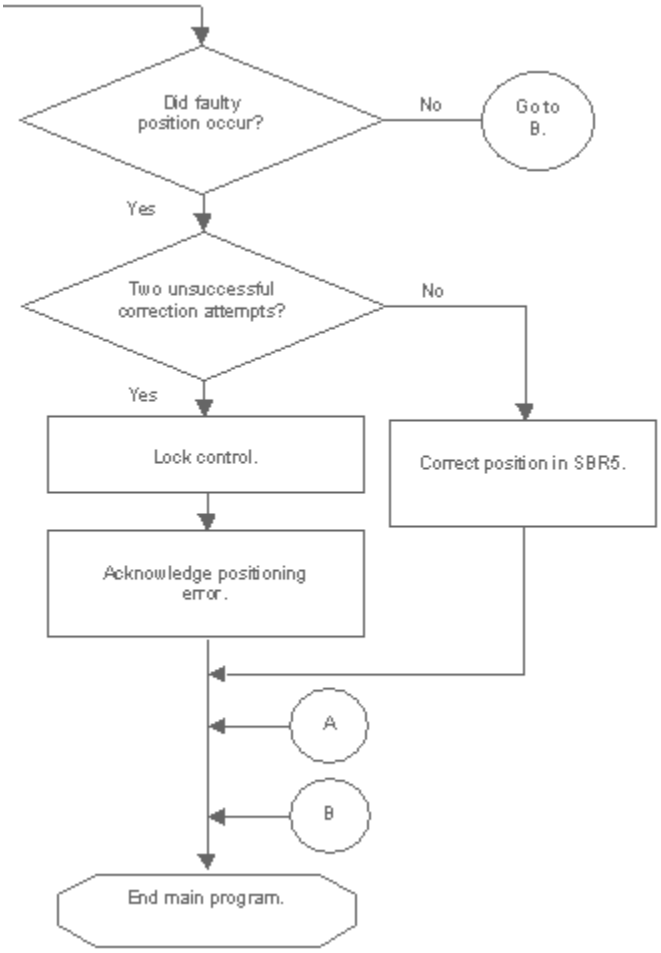

```
//Main, cài đặt HSC0, gọi chương trình con 0 và 1
Network1
LD SM0.1
R Q0.0,1
MOVB 16#F8, HSC0_Ctrl
MOVD 0, HSC0_CV
MOVD 1000, HSC0_PV
HDEF 0.0
CALL SBR_0
CALL SBR_1
//SBR0, KHỞI ĐỘNG VÀ CHO PHÉP PTO
NETWORK1
LD SM0.0
MOVB 16#8D, PLS0_Ctrl
MOVW 1, PLS0_Cycle // bề rộng xung 1ms
MOVD 30000, PTO0_PC //Số xung là 30000
PLS 0 // PHÁT XUNG Ở Q0.0
//SBR1, khởi động HSC0
NETWORK1
LD SM0.0
ATCH INT0, 12
ENI
HSC 0
```

```
//INT0 Set Q0.1 và nạp giá trị đặt cho HSC0
LD SM0.0
S Q0.1, 1
MOVB 16#A0, HSC0_Ctrl
MOVD 1500, HSC0_PV
ATCH INT1, 12
HSC0
//INT1 SET Q0.2 , đếm xuống và nạp giá trị đặt mới
LD SM0.0
S Q0.2, 1
MOVB 16#B0, HSC_Ctrl
MOVD 1000,HSC0_PV
ATCH INT2, 12
HSC0
//INT2, Reset Q0.1, Q0.2, đếm lên, reset giá trị hiện tại
NETWORK1
LD SM0.0
R Q0.1, 2
MOVB 16#D8, HSC0_Ctrl
MOVD 0, HSC0_CV
ATCH INT0, 12
HSC0
```

Ví dụ: điều khiển vị trí động cơ bước




Start main program.
Execute initializations (only on first scan cycle).
Choose rotation direction (I1.5).
Specify operating mode (search for reference/positioning) in SBR1.
Reference point set?
No
Yes
Search for reference.
Set reference point.
Go to A.
Set position in SBR2.
Monitor position.




```
//Main
Network 1 //  Specify Pulse Width and Define High-Speed Counter HSC2
LD     SM0.1
ATCH   INT0, 19
ENI
MOVW   +0, SMW70        // Specify pulse width of 0.
CALL   SBR6
MOVB   16#FC, SMB57
HDEF   2, 10
HSC    2
Network 2 //  Check Positioning and Specify Cycle Time
//  Load memory word MW25 to check if there is a positioning error. If the value in MW25 is equal to 0 (no positioning error),  move the constant 200 to special memory word SMW68 to specify a cycle time of 200 microseconds for pulse train output Q0.0.
LDW=   MW25, +0
```

```
MOVW   +200, SMW68

Network 3 //  Enable Counterclockwise Rotation
// The switch at input I1.5 specifies the direction of the motor rotation. If input I1.5 is set (signal = 1), the motor runs counterclockwise//
LDN    M0.1
A      I1.5
S      Q0.2, 1
Network 4 //  Enable Clockwise Rotation
// The switch at input I1.5 specifies the direction of the motor rotation. If input I1.5 is not set (signal = 0), the motor runs clockwise//
LDN    M0.1
AN     I1.5
R      Q0.2, 1
Network 5 //  Activate Blocking
// To protect personnel and equipment, the program provides the opportunity to block the drive after activating the motor STOP switch at input I1.1 or after three faulty positionings have been reported at memory word MW25 (value in MW25 = 3). Either event sets the blocking memory bit at M0.2 and turns the drive off  immediately.//
LD     I1.1
OW=    MW25, +3
S      M0.2, 1
Network 6 //  Cancel Blocking
// To prevent the uncontrolled start of the drive when the STOP switch is released or the number of faulty positioning reports drops below three, M0.2 is reset only if both of the switches at input I1.0 (motor START) and input I1.1 (motor STOP) are released and the value in memory word MW25 is less than or equal to two.//
LDN    I1.1
AN     I1.0
AW<=   MW25, +2
R      M0.2, 1
Network 7 //  Call Subroutine SBR1 to Specify Operating Mode
// If the switch at input I1.4 (set reference point) is activated and there is no blocking active and the first positioning has not been set, the program calls subroutine SBR1 to specify the operating mode.//
LD     I1.4
EU
AN     M0.2
AN     M0.4
CALL   SBR1
Network 8 //  Cài đặt HSC, PWM/PTO , SMD58 là giá trị mới cho HSC2, SMB67 khởi động PTO
LD     I1.0
EU
```

```
AN    M0.1
AN    M0.2
AN    M0.4
AD>=  SMD72, +1
MOVD  +0, SMD58
HSC   2
MOVB  16#85, SMB67
S     M0.1, 1
PLS   0
```

Network 9 // Call Subroutine SBR2

//If a reference point has been specified (M0.3 is set) and no positioning has happened yet (M0.4 is not set), the program calls SBR2 to calculate the number of steps.

```
LD    M0.3
AN    M0.4
CALL  SBR2
```

Network 10 // Start Position Correction Wait Timer

// Once the wait timer T97 has expired, if a positioning fault is detected, a second wait timer (T98) is started.

```
LD    M1.1
AW>=  MW25, +1
 AN   M0.2
TON   T98, +100
```

Network 11 // Position Correction

// After timer T98 has expired, call subroutine SBR5 to calculate the steps necessary to correct the positioning. Move the constant 1000 to special memory word SMW68 to specify a cycle time of 1 ms for pulse train output Q0.0.Move the hexadecimal value 85 (binary 1000 0101) to special memory byte SMB67. This enables the update of the cycle time and the PTO pulse count value, sets the time base to 1 microsecond/tick, selects PTO mode, and enables output Q0.0. Set memory bit M0.1 (drive ON). Examine the special memory bits for pulse output Q0.0 and enable the pulse function as defined by those bits. Reset memory bit M1.1.

```
//
LD    T98
CALL  SBR5                  .
MOVW  +1000, SMW68
MOVB  16#85, SMB67     //Load the control bits for pulse train output Q0.0.
S     M0.1, 1
PLS   0
 R    M1.1, 1
```

Network 12 // Position Monitoring

// Memory bit M20.0 is set when the pulse output at output Q0.0 is completed. If the pulse output is completed (M20.0 = 1) and there is no blocking (M0.2 = 0), start on-delay timer T97 with a preset time (PT) of 50 (500 ms).

```
LD     M20.0
AN     M0.2
TON    T97, +50
Network 13 //  Waiting Time Expires: Call Subroutine SBR4
//  After on-delay timer T97 expires, set memory bit M1.1 and call subroutine SBR4 for position monitoring. Reset memory bit M20.0 to allow the pulse output to restart and to reset timer T97.
LD     T97
S      M1.1, 1
CALL   SBR4
R      M20.0, 1
Network 14 //  Stop Drive and Call Subroutine SBR0
//  Load input I1.1 to stop the motor when the switch is activated.
LD     I1.1
EU
A      M0.1
CALL   SBR0
Network 15 //  Call Subroutine SBR6 after Three Positioning Errors
//If the switch at input I1.4 (set or delete reference point) is activated and three positioning errors have been reported, the program calls SBR6 to reinitialize memory bits, counters, and displays.
LD     I1.4
EU
AW=    MW25, +3
CALL   SBR6

// SBR0,  Stop Drive and Set PWM/PTO Control for Output Q0.0
//  In this network, special memory byte SMB67 initializes the pulse
//  width modulation function of output Q0.0.
LD     SM0.0
MOVB   16#CB, SMB67
PLS    0
R      M0.1, 1
// SBR1,  Call Subroutine SBR0 to Stop the Drive
Network 1
LD     M0.1
CALL   SBR0
Network 2 //  Request Reference Point Trend
LD     M0.3
R      M0.3, 1
R      Q1.0, 1
MOVD   16#1999997C, SMD72
```

```
CRET
Network 3 //  Set Reference Point and Display Message
LDN   M0.3
S     M0.3, 1
S     Q1.0, 1
// SBR2,  Controlled Positioning
Network 1
```

// In this subroutine, the controller reads the positioning angle in binary coded decimal format from the input byte IB0 into the memory byte MB11. The pulse number which corresponds to this angle is calculated by dividing the angle to be traversed in degrees by 360 degrees and then multiplying that result by the number of steps per motor revolution.

// The stepper motor used in this sample application is applied in half-step operation, with the number of steps per motor revolution = 1000. To reduce the size of the numbers to be stored while retaining the correct ratio, 1000/360 = 25/9. The calculated number of steps is subsequently rounded off in subroutine SBR3.//

```
LD    SM0.0
MOVB  IB0, MB11
R     M8.0, 24
MOVW  +9, VW10
DIV   VW10, MD8
MOVW  MW8, MW14
MUL   +25, MD8
MUL   +25, MD12
DIV   VW10, MD12
CALL  SBR3                .
MOVW  +0, MW12
+D    MD12, MD8
MOVD  MD8, AC2
MOVD  AC2, SMD72

Network 2 // Tính phụ 2 khi quay ngược
LD    I1.5
INVD  AC2
INCD  AC2
Network 3 // Đặt giới hạn sai số
LD    SM0.0
MOVD  AC2, AC0
MOVD  AC2, AC1
-D    +2, AC0
+D    +2, AC1
// SBR3, Làm tròn số bước
```

```
Network 1
LDW>=  MW12, +5
INCW   MW14
// SBR4,  Nếu định vị đúng, xóa bit báo hiệu Q1.1
Network 1
LDD<=  HC2, AC1
AD>=   HC2, AC0
R      M0.4, 1
R      M25.0, 16
R      Q1.1, 1
CRET
Network 2 // Nếu sai lệch vị trí thì báo
LD     SM0.0
INCW   MW25
S      Q1.1, 1
// SBR5, Tính sai lệch vị trí
Network 1
LD     SM0.0
MOVD   AC2, AC3
-D     HC2, AC3
Network 2
LDD<=  HC2, AC0
MOVD   AC3, SMD72
Network 3 //  Khi quay ngược đến giới hạn dưới thì quay ngược
LDD<=  HC2, AC0
A      I1.5
R      Q0.2, 1
CRET
Network 5 //  Đảo ngược và tăng số bước
LDD>=  HC2, AC1
INVD   AC3
INCD   AC3
MOVD   AC3, SMD72
Network 6 //  Khi đạt đến giới hạn trên thì đảo chiều .
LDD>=  HC2, AC1
AN     I1.5
S      Q0.2, 1
// SBR   6, khởi động lại
Network 1
LD     SM0.0
R      M0.0, 128
```

```
R     M25.0, 16
R     Q1.0, 2
MOVD  16#1999997C, SMD72
//INT0
```

Network 1 // INT0., Reset Drive ON M0.1 Bit và Set Pulse Output Complete Bit M20.0

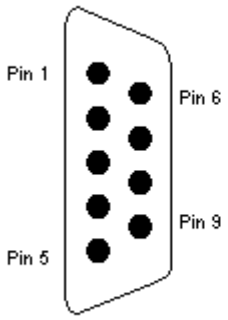
```
LD    SM0.0
R     M0.1, 1
S     M20.0, 1
```

Network 2 // Set First Positioning Bit M0.4

```
LDN   M0.4
S     M0.4, 1
```

9.29 TRUYỀN THÔNG

PLC có một hoặc hai cổng (Port 0, Port 1) dùng để truyền thông nối tiếp RS485.



Pin Number	Signal
1	Shield
2	24 V Return
2	RS-485 Signal B
4	Request-to-Send
5	5 V Return
6	+5 V
7	+24 V
8	RS-485 Signal A
9	Not applicable

Thông qua cổng nối tiếp có thể ghép S7-200 với các thiết bị khác như S7-200, S7-300, HMI, PC. Muốn sử dụng cổng nối tiếp theo giao thức người dùng, ta phải đặt cổng ở chế độ freeport bằng cách đặt nội dung cho SMB30 (port 0) và SMB130 (port 1) theo bảng sau:

Bảng 9.6

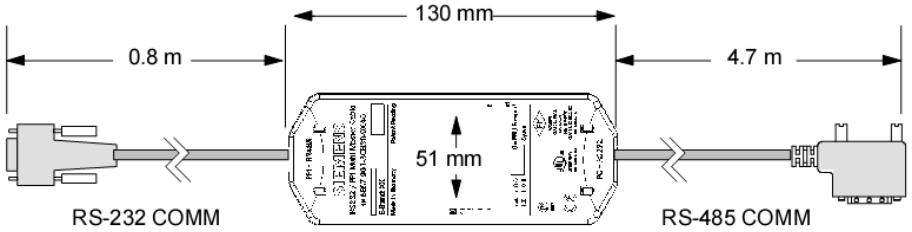
SMB 30, SMB 130	Miêu tả
Bit 7, 6	00, 10: không parity
	01: parity chẵn
	11: parity lẻ
Bit 5	0: data 8 bit

	1: data 7 bit
Bit 4, 3, 2	000: 38400 bps (CPU212 19200)
	001: 19200
	010: 9600
	011: 4800
	100: 2400
	101: 1200
	110: 115200 (CPU212 600)
	111: 57600 (CPU212 300)
Bit 1, 0	00, 11: PPI slave (đùng cho mạng)
	01: Freeport
	10: PPI master (đùng cho mạng)

Như vậy với giao thức free port, 9600 baud ta nạp 9 vào SMB30 và 5 với 19200 baud

Ngoài chế độ truyền thông freeport mà giao thức do người dùng qui định, PLC còn dùng chế độ PPI master /slave, giao thức truyền thông đã qui định sẵn bởi phần mềm trong PLC.

Kết nối PLC với thiết bị khác thông qua cable kết nối PPI Multi Master, có hai loại RS232 và USB, hay cable PPI/PC

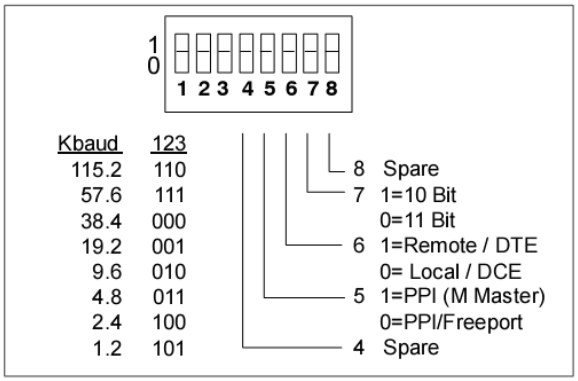


RS-485 Connector Pin-out		RS-232 Local Connector Pin-out	
Pin Number	**Signal Description**	**Pin Number**	**Signal Description**
1	No connect	1	Data Carrier Detect (DCD) (not used)
2	24 V Return (RS-485 logic ground)	2	Receive Data (RD) (output from PC/PPI cable)
3	Signal B (RxD/TxD+)	3	Transmit Data (TD) (input to PC/PPI cable)
4	RTS (TTL level)	4	Data Terminal Ready (DTR)[1]
5	No connect	5	Ground (RS-232 logic ground)
6	No connect	6	Data Set Ready (DSR)[1]
7	24 V Supply	7	Request To Send (RTS) (not used)
8	Signal A (RxD/TxD-)	8	Clear To Send (CTS) (not used)
9	Protocol select	9	Ring Indicator (RI) (not used)

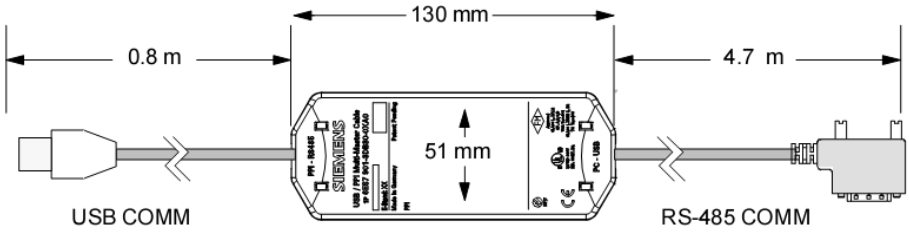
[1] Pins 4 and 6 are connected internally.

Khi nối cáp với máy tính, đặt switch 7 vị trí 0, switch 6 vị trí local (0), còn khi nối với modem, đặt switch 6 vị trí remote (1), máy

tính là Master nên chọn switch 5 vị trí 1, nếu chọn vận tốc truyền 9600 thì vị trí các switch là 01001000

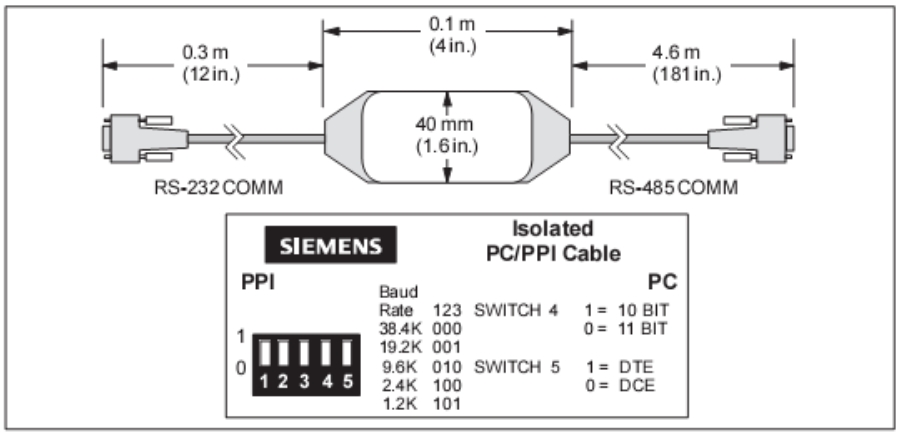


LED	Color	Description
Tx	Green	RS-232 transmit indicator
Rx	Green	RS-232 receive indicator
PPI	Green	RS–485 transmit indicator

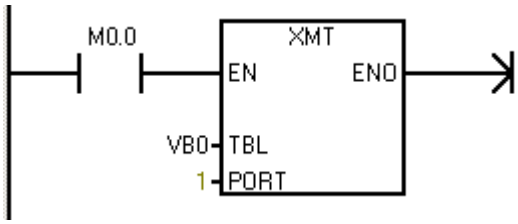


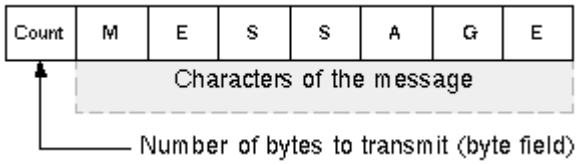
LED	Color	Description
Tx	Green	USB transmit indicator
Rx	Green	USB receive indicator
PPI	Green	RS-485 transmit indicator

RS-485 Connector Pin-out		USB Connector Pin-out	
Pin Number	Signal Description	Pin Number	Signal Description
1	No connect	1	USB – DataP
2	24 V Return (RS-485 logic ground)	2	USB – DataM
3	Signal B (RxD/TxD+)	3	USB 5V
4	RTS (TTL level)	4	USB logic ground
5	No connect		
6	No connect		
7	24 V Supply		
8	Signal A (RxD/TxD–)		
9	Protocol select (low = 10 bit)		



Tương tự PLC Omron, ở chế độ freeport có thể truyền và nhận data bằng hai lệnh XMT và RCV. Khi dùng lệnh XMT, dữ liệu truyền lấy từ bảng có địa chỉ đầu ở TABLE, còn dữ liệu nhận chứa vào TABLE, chiều dài bảng tối đa 255. Byte đầu của bảng cho biết chiều dài dữ liệu. Khi ký tự cuối của bảng được gởi sẽ báo sự kiện ngắt 9 hay 26 hay tác động SM4.5 (port 0), SM4.6 (port 1), hai bit này ON khi đã thực hiện xong việc truyền và OFF khi đang truyền.





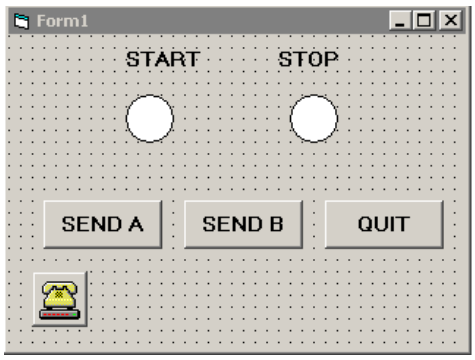
Sự kiện ngắt 8 (Port 0), 25 (Port 1) sẽ xảy ra khi thu một ký tự, lúc này SMB2 chứa ký tự vừa nhận còn SM3.0 chứa kết quả kiểm tra parity (0: không có lỗi parity, 1: có lỗi parity).

Chú ý là không cùng lúc nhận và gởi dữ liệu qua port, giữa hai chế độ cần có khoảng nghỉ.

Ví dụ: Truyền dùng lệnh XMT và nhận ký tự dùng SMB2, máy tính gởi xuống ký tự A thì Q0.0 ON, gởi ký tự B thì Q0.0 OFF, khi I0.0 có cạnh lên thì PLC gởi lên ký tự A, I0.1 có cạnh lên thì PLC gởi lên ký tự B, máy tính nhận được A thì đèn Start sáng, máy tính nhận được B thì đèn Stop sáng, Có thời gian nghỉ 5ms từ khi PLC nhận dữ liệu đến khi gởi lên

```
//Chương trình chính
//Network 1
LD     SM0.1
MOVB   9, SMB30
ATCH   INT_0, 8
ENI
MOVB   'B', VB103
MOVB   'A', VB101
MOVB   1, VB102
MOVB   1, VB100
// Network2
LDN    M1.0
A      I0.0
EU
XMT    VB100, 0
S      Q0.1, 1
// Network3
LDN    M1.0
A      I0.1
EU
XMT    VB102, 0
R      Q0.1, 1
//Chương trình ngắt0
//Network1
LD  SM0.0
LPS
S M1.0,1
AB=    SMB2, 'A'
S      Q0.0, 1
LPP
AB=    SMB2, 'B'
R      Q0.0, 1
//Network2
LD     M1.0
MOVB   5, SMB34
ATCH   INT1, 10
//Chương trình ngắt1
LD     SM0.0
R      M1.0, 1
DTCH   10
```

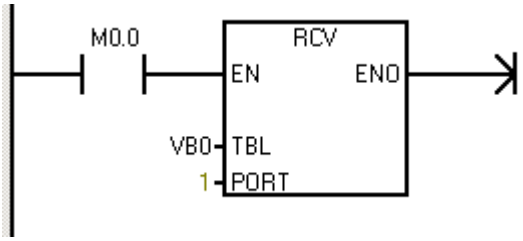
Chương trình VB

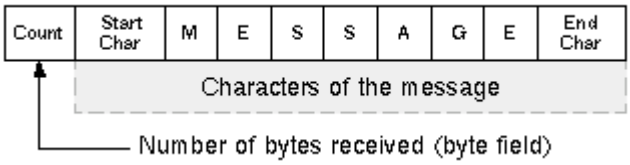



```
Private Sub cmdQuit_Click()
  MSComm1.PortOpen = False
  End
End Sub

Private Sub cmdSendA_Click()
  MSComm1.Output = "A"
End Sub

Private Sub cmdSendB_Click()
  MSComm1.Output = "B"
End Sub

Private Sub Form_Load()
  If MSComm1.PortOpen Then MSComm1.PortOpen = False
  With MSComm1
        CommPort = 3
.       Settings = "9600,n,8,1"
        InputLen = 0
      . InputMode = 0
.       RThreshold = 1
.       PortOpen = True
  End With
  With Shape1
.       FillColor = vbWhite
.       Shape = 3
.       FillStyle = 0
  End With
  With Shape2
.       FillColor = vbWhite
.       Shape = 3
.       FillStyle = 0
        End With
End Sub

Private Sub MSComm1_OnComm()
  Dim data
  If MSComm1.CommEvent = comEvReceive Then
        data = MSComm1.Input
  Select Case data
```

```
                Case "A"
                Shape1.FillColor = vbBlue
                Shape2.FillColor = vbWhite
                Case "B"
                Shape1.FillColor = vbWhite
                Shape2.FillColor = vbRed
    End Select
End If
End Sub
```

Lệnh RCV dùng để nhận một bản tin chứa vào một bảng, ta phải qui định các điều kiện khởi đầu và kết thúc cho bản tin và dùng các ô nhớ SMB86..SMB94 (Port 0), SMB186..SMB194 (Port 1) hỗ trợ, Khi thu xong bằng lệnh RCV sẽ tác động sự kiện ngắt 23 Port 0 (24 Port 1) . Chú ý là hai lệnh XMT và RCV không thể sử dụng đồng thời, nếu hai lệnh thực hiện cùng lúc thì lệnh sau không thực hiện và ENO là 0.





S7-200 Symbol Name	SM Address		
	Port 0	Port 1	
P0_Start_Char	SMB88		Start of message character.
P1_Start_Char		SMB188	
P0_End_Char	SMB89		End of message character.
P1_End_Char		SMB189	
P0_Idle_Time	SMW90		Word data: Idle line time period given in milliseconds. The first character received after the idle line time has expired is the start of a new message.
P1_Idle_Time		SMW190	
P0_Timeout	SMW92		Word data: Inter-character/message timer timeout value given in milliseconds. If the time period is exceeded, the receive message is terminated.
P1_Timeout		SMW192	
P0_Max_Char	SMB94		Maximum number of characters to be received (1 to 255 bytes).
P1_Max_Char		SMB194	**Note:** This range must be set to the expected maximum buffer size, even if the character count message termination is not used.

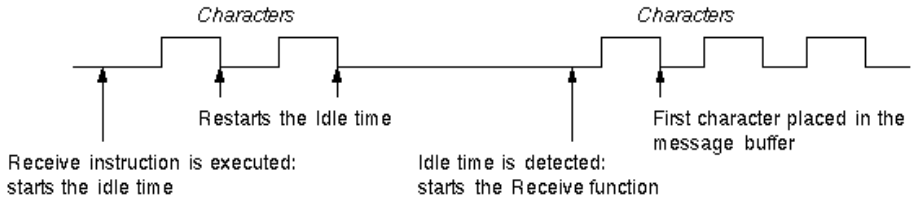
			Receive Message Control Byte								
			MSB								LSB
	Port 0	Port 1	Bit Format	7							0
P0_Ctrl_Rcv	SMB87			en	sc	ec	il	c/m	Tmr	bk	0
P1_Ctrl_Rcv		SMB187									
P0_Ctrl_Rcv_7	SM87.7		en:	0	= receive message function is disabled						
P1_Ctrl_Rcv_7		SM187.7		1	= receive message function is enabled						
P0_Ctrl_Rcv_6	SM87.6		sc:		0	= ignore SMB88 or SMB188					
P1_Ctrl_Rcv_6		SM187.6			1	= use the value of SMB88 or SMB188 to detect start of message					
P0_Ctrl_Rcv_5	SM87.5		ec:			0	= ignore SMB89 or SMB189				
P1_Ctrl_Rcv_5		SM187.5				1	= use the value of SMB89 or SMB189 to detect end of message				
P0_Ctrl_Rcv_4	SM87.4		il:				0	= ignore SMW90 or SMB190			
P1_Ctrl_Rcv_4		SM187.4					1	= use the value of SMW190 to detect an idle line condition			
P0_Ctrl_Rcv_3	SM87.3		c/m:					0	= timer is an inter-character timer		
P1_Ctrl_Rcv_3		SM187.3						1	= timer is a message timer		
P0_Ctrl_Rcv_2	SM87.2		tmr:						0	= ignore SMW92 or SMW192	
P1_Ctrl_Rcv_2		SM187.2							1	= terminate receive if the time period in SMW92 or SMW192 is exceeded	
P0_Ctrl_Rcv_1	SM87.1		bk:							0	= ignore break conditions
P1_Ctrl_Rcv_1		SM187.1								1	= use break condition as start of message detection

S7-200 Symbol Name	SM Address		Receive Message Status Byte								
			MSB								LSB
	Port 0	Port 1	Bit Format	7							0
P0_Stat_Rcv	SMB86			n	r	e	0	0	t	c	P
P1_Stat_Rcv		SMB186									
P0_Stat_Rcv_7	SM86.7		n:	1	= Receive message was terminated by user disable command						
P1_Stat_Rcv_7		SM186.7									
P0_Stat_Rcv_6	SM86.6		r:		1	= Receive message terminated: error in input parameters or missing start or end condition					
P1_Stat_Rcv_6		SM186.6									
P0_Stat_Rcv_5	SM86.5		e:			1	= End character received				
P1_Stat_Rcv_5		SM186.5									
P0_Stat_Rcv_2	SM86.2		t:					1	= Receive message terminated: timer expired		
P1_Stat_Rcv_2		SM186.2									
P0_Stat_Rcv_1	SM86.1		c:						1	= Receive message terminated: maximum character count achieved	
P1_Stat_Rcv_1		SM186.1									
P0_Stat_Rcv_0	SM86.0		p:							1	= Receive message was terminated because of a parity error
P1_Stat_Rcv_0		SM186.0									

Chú ý là thay đổi cấu hình chỉ hiệu quả khi thực hiện lệnh RCV. Khi lệnh RCV đã gây ngắt thì ngừng hoạt động

Để nhận biết bắt đầu bản tin có thể dùng nhiều cách:

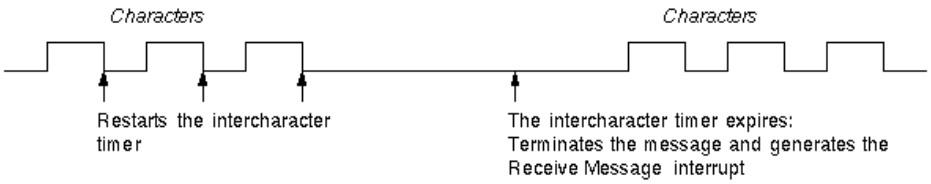
- Qui định ký tự bắt đầu trong ô nhớ SMB88 (SMB188), trong byte điều khiển cho bit sc SM87.6 (SM187.6) logic 1 (thường dùng khi truyền ký tự)
- Phát giác đường dây rỗi: qui định thời gian đường dây rỗi giữa hai lần phát bản tin, thường là hơn ba lần thời gian truyền một byte, trong ô nhớ SMW90 (SMW190) đơn vị là ms. Khi lệnh RCV được thực hiện PLC sẽ tính thời gian đường dây rỗi, các byte nhận trước thời gian này sẽ làm khởi động lại bộ tính thời gian này, sau thời gian rỗi qui định mới chấp nhận các byte đưa vào bảng (thường dùng khi truyền nhị phân) trong byte điều khiển cho bit il SM87.4 (SM187.4) lên 1



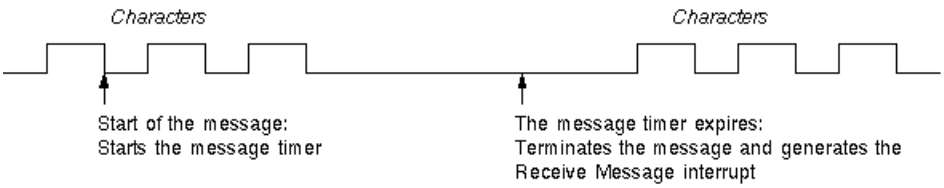
- Phát giác tín hiệu break: đường truyền ở điều kiện break khi nó ở mức zero hơn thời gian truyền một byte, lúc đó PLC sẽ nhận các ký tự sau điều kiện break, bit 1 (bk) của byte điều khiển đặt lên 1

Để nhận biết kết thúc bản tin gây ra ngắt có thể dùng:

- Ký tự cuối trong ô nhớ SMB89 (SMB189), bit 5 (ec) byte điều khiển logic 1
- Timer giữa ký tự: nếu thời gian giữa hai ký tự lớn hơn thời gian trong ô nhớ SMW92 (SMW192) đơn vị ms thì coi như kết thúc truyền bản tin, bit 2 (tmr) của byte điều khiển mức 1, bit 3 (c/m) mức 0



- Timer bản tin: qui định thời gian truyền một bản tin trong ô nhớ SMW92 (SMW192) , sau thời gian này coi như kết thúc nhận, bit 2 (tmr) của byte điều khiển mức 1, bit 3 (c/m) mức 1



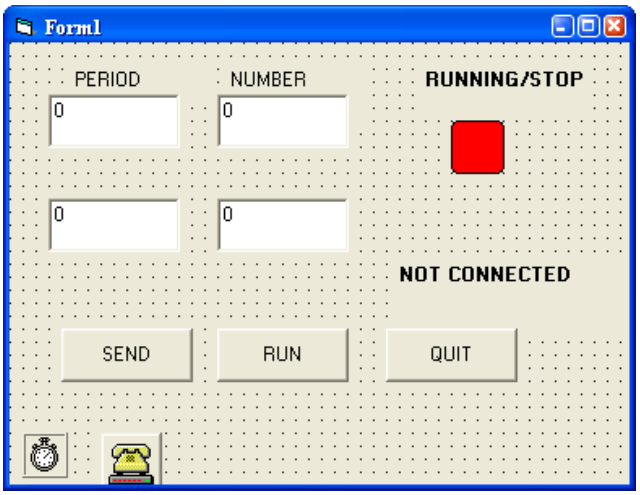
- Số ký tự tối đa: qui định truyền tối đa bao nhiêu ký tự SMB94 (SMB194), nếu nhận quá gây ra ngắt
- Sai parity
- Do kết thúc thu, đặt bit 7 (en) của byte điều khiển về 0

Ví dụ: nhận bản tin và truyền trở lại bản tin đã nhận dùng lệnh RCV và XMT

```
NETWORK 1 // Main Program
LD SM0.1 //On the first scan,
MOVB 16#09 SMB30 //Initialize Freeport:
 // - Select 9600 baud
 // - Select 8 data bits
 // - Select no parity
MOVB 16#B0 SMB87 //Initialize RCV message control byte:
 // RCV enabled, detect end of message character, detect idle line condition as message start condition
MOVB 16#0A SMB89 //Set end of message character to hex 0A (line feed)
MOVW +5 SMW90 //Set idle line timeout to 5 ms.
MOVB 100 SMB94 //Set maximum number of characters to 100.
ATCH INT_0 23 //Attach interrupt 0 to the receive complete event.
ATCH INT_2 9 //Attach interrupt 2 to the transmit complete event.
ENI
RCV VB100 0 //Enable receive box with buffer at VB100 for port 0
NETWORK 1 // Interrupt 0
//Receive complete interrupt routine
LDB= SMB86 16#20 //If receive status shows receive of end character,
MOVB 10 SMB34 //then attach a 10 ms timer
ATCH INT_1 10 //to trigger a transmit
CRETI //and return.
NOT
RCV VB100 0 //If the receive completed for any other reason, then start a new receive.
NETWORK 1 // Interrupt 1
//10 ms timer interrupt
LD SM0.0
DTCH 10 //Detach timer interrupt
XMT VB100 0 //Transmit message back to user on port 0
NETWORK 1 // Interrupt 2
LD SM0.0
RCV VB100 0
```

Ví dụ: Viết chương trình PLC nhận dữ liệu nối tiếp dạng mã ASCII, baudrate 19200. Lệnh thứ nhất ký tự "A" sau đó là **sáu byte**, sẽ được PLC cất vào **ba ô nhớ** từ VB0, dùng làm chu kỳ xung và số lượng xung cho lệnh phát xung PTO ở Q0.0. Lệnh thứ hai "BC" cho M0.0 ON.. Khi nhận được dữ liệu PLC gởi trả lại thông tin đã nhận, nếu M0.0 chuyển trạng thái từ OFF sang ON thì PLC cho M0.0 OFF và phát xung, phát đủ xung gởi lên máy tính ký tự '@'. Viết chương trình VB gởi và nhân dữ liệu với PLC như hình dưới. Trong ô txtPeriod gõ hai số hex, ô txtNumber gõ bốn số hex, bầm cmdSend gởi lệnh thứ

nhất, bấm nút cmdRun gởi lệnh thứ hai. Khi PLC gởi trả lại lệnh thứ nhất hiển thị sáu byte trong hai ô text dạng số nguyên, khi PLC gởi trả lại lệnh thứ hai cho đèn shpRun màu xanh, khi PLC gởi "@" cho đèn này màu đỏ. Khi máy tính gởi lệnh, nếu sau 0.1s không nhận được trả lời từ PLC thì báo nhãn lblError "NOT CONNECTED"




```
Private Declare Function Beep Lib "kernel32" (ByVal dwFreq As Long, ByVal dwDuration As Long) As Long

Dim datain
Dim received
Dim timeout

Private Sub cmdQuit_Click()
        MSComm1.PortOpen = False
        End
End Sub

Private Sub cmdRun_Click()
        lblError.Visible = False
        datain = ""
        Shape1.FillColor = vbRed
        MSComm1.Output = "BC"
        Timer1.Enabled = True
        received = False
End Sub

Private Sub cmdSend_Click()
        lblError.Visible = False
        datain = ""
        MSComm1.InBufferCount = 0
        Text3.Text = ""
        Text4.Text = ""
```

```
        datain = ""
        MSComm1.Output = "A" & Text1.Text & Text2.Text
        cmdRun.Enabled = True
        Timer1.Enabled = True
        received = False
End Sub

Private Sub Form_Load()
        With MSComm1
        .CommPort = 3
        .Settings = "19200,n,8,1"
        .RThreshold = 1
        .InputLen = 0
        .PortOpen = True
        End With
        lblError.Visible = False
        Timer1.Interval = 500
        Timer1.Enabled = False
        Text1.Text = F0
        Text2.Text = 0056
        cmdRun.Enabled = False
        received = False
        datain = ""
End Sub

Private Sub MSComm1_OnComm()
        If MSComm1.InBufferCount > 0 Then
                received = True
                datain = datain & MSComm1.Input
        End If
        Text5.Text = datain
        If (Mid(datain, 1, 2) = "BC") Then
                Shape1.FillColor = vbBlue
                cmdRun.Enabled = False
                datain = ""
        End If
        If (Mid(datain, 1, 1) = "A") And Len(datain) > 6 Then
                Text3.Text = Mid(datain, 2, 2)
                Text4.Text = Mid(datain, 4, 4)
                datain = ""
        End If
        If Mid(datain, 1, 1) = "@" Then
                Shape1.FillColor = vbRed
                cmdRun.Enabled = True
                Beep 2000, 500
                datain = ""
        End If
End Sub

Private Sub Timer1_Timer()
        If Not (received) Then
                lblError.Visible = True
                Beep 1000, 200
```

```
                timeout = True
        End If
        Timer1.Enabled = False
End Sub
```

Chương trình PLC

Chương trình chính MAIN

```
Network 1
LD    SM0.1
LPS
MOVB  5, SMB30
ATCH  INT1, 19
AENO
ATCH  INT0, 9
AENO
ATCH  INT2, 23
LRD
ENI

MOVB  2#10011101, SMB67
AENO
MOVB  2#10010100, SMB87
AENO
MOVB  100, SMB94
LRD
AENO
MOVB  1, VB20
AENO
MOVB  '@, VB21
LRD
MOVB  0, SMB68
AENO
MOVW  50, SMW90
AENO
MOVW  20, SMW92
LRD
MOVD  0, VD10
AENO
MOVD  0, VD14
AENO
MOVW  0, SMW72
LPP
RCV   VB10, 0
```

Ngắt Thu INT4

```
Network 1
LD    SM0.0
XMT   VB10, 0
=     M0.1
=     Q0.1

Network 2
LDB=  VB10, 7
ATH   VB12, VB0, 6
MOVB  VB0, SMB69
MOVW  VW1, SMW74

Network 3
LDB=  VB10, 2
R     Q0.0, 1
PLS   0
S     M0.0, 1
```

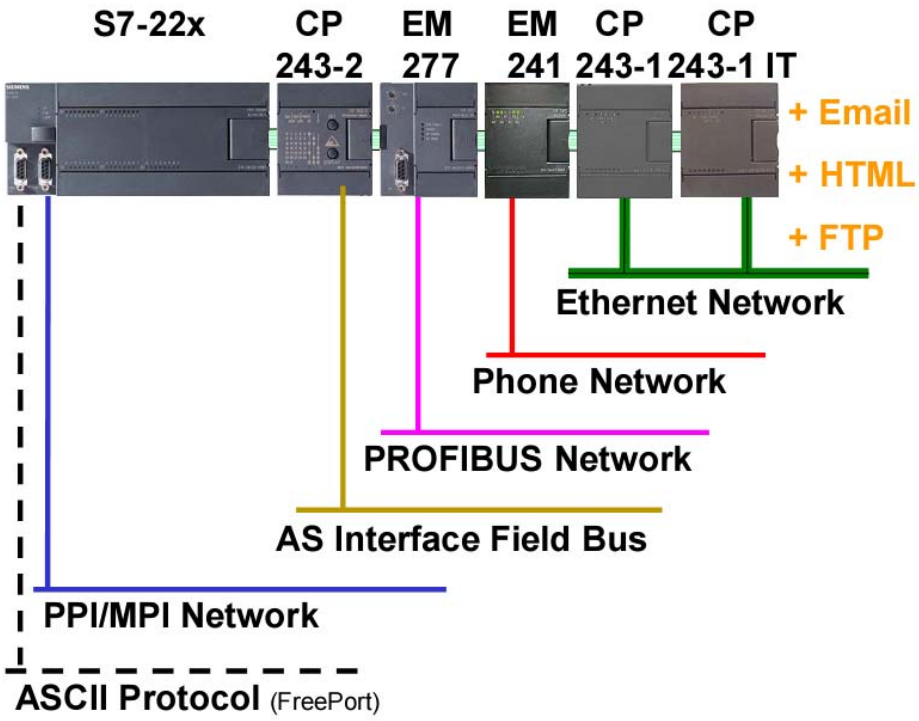
Ngắt Truyền INT0

```
Network 1
LDN   M0.0
RCV   VB10, 0
```

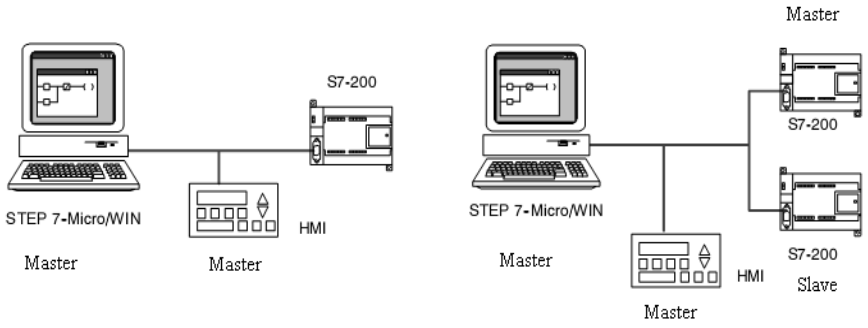
Ngắt PTO INT1

```
Network 1 /
LD     SM0.0
R      M0.0, 1
XMT    VB20, 0
```

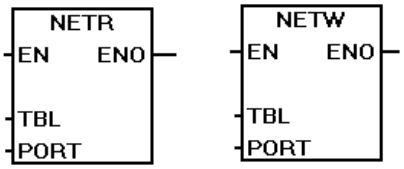
9.30 LỆNH MẠNG



Khi nối nhiều PLC S7-200 với nhau qua mạng 485 PPI ta dùng lệnh NETR/ NETW để đọc/ghi dữ liệu, các PLC phải có địa chỉ cụ thể. Cấu hình mạng PPI gồm có PLC master và các slave, nếu có màn hình hay máy tính thì các thiết bị này cũng là master, lệnh NETR/ NETW đặt trong chương trình của PLC master, dữ liệu truyền nhận giữa master và slave chứa trong một bảng, trong một chương trình có tối đa 8 lệnh NETW hay /và NETR cùng được kích hoạt. Lệnh NETW/R sử dụng một bảng để truy cập slave, có thể ghi /đọc đến 16 byte dữ liệu mỗi slave



Giao tiếp mạng PPI



Byte Offset	7				0
0	D	A	E	0	Error code
1	Remote station address				
2	Pointer to the data				
3	area in the				
4	remote station				
5	(I, Q, M, or V)				
6	Data length				
7	Data byte 0				
8	Data byte 1				
	⋮				
22	Data byte 15				

D: done, khi lệnh được thực hiện thì ở mức 1

A: active, lệnh đang chờ được thực hiện trong hàng đợi

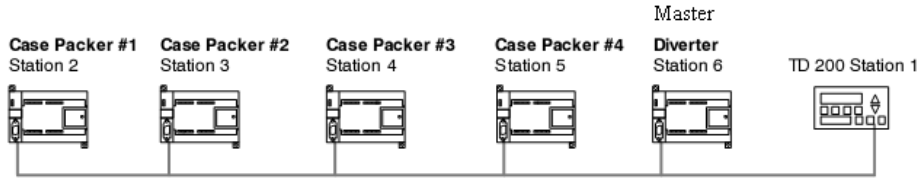
E: error, báo có lỗi, mã lỗi trong error code

Error code:

- 1: trạm remote không trả lời
- 2: lỗi thu (sai parity, frame…)
- 3: trùng địa chỉ, lỗi phần cứng

- 4: hàng đợi tràn, quá 8 lệnh đang chờ
- 5: dùng lệnh NETW/R nhưng không cài đặt PPI master
- 6: tham số bảng không đúng
- 7: Trạm remote bận
- 8: Sai giao thức
- 9: Sai địa chỉ hay kích thước dữ liệu

Ví dụ: một dây chuyền gồm 5 PLC, PLC master có địa chỉ 6, các PLC slave điều khiển các máy đóng gói có địa chỉ lần lượt 2, 3, 4, 5. PLC master gởi hai byte dữ liệu xuống địa chỉ VW101 của slave và nhận ba byte từ địa chỉ VB100. Bảng cho lệnh NETR ở các địa chỉ VB200, VB210, VB220, VB230. Bảng cho lệnh NETW ở các địa chỉ VB300, VB310, VB320, VB330.



Bảng thu dùng đọc dữ liệu trạm 2

	7				0
VB200	D	A	E	0	Error Code
VB201	Remote station address = 2				
VB202	Pointer to the				
VB203	data area				
VB204	in the				
VB205	Remote station = (&VB100)				
VB206	Data length = 3 bytes				
VB207	Control				
VB208	Status (MSB)				
VB209	Status (LSB)				

Bảng phát dùng gởi dữ liệu đến trạm 2

	7				0
VB300	D	A	E	0	Error Code
VB301	Remote station address = 2				
VB302	Pointer to the				
VB303	data area				
VB304	in the				
VB305	Remote station = (&VB101)				
VB306	Data length = 2 bytes				
VB307	0				
VB308	0				

Sau đây là đoạn chương trình đọc và gởi dữ liệu cho trạm 2

NETWORK 1

LD SM0.1 // Chu kỳ quét đầu,

MOVB 2 SMB30 // cho phép mode PPI master

FILL +0 VW200 68 // xóa đệm phát và thu

NETWORK 2

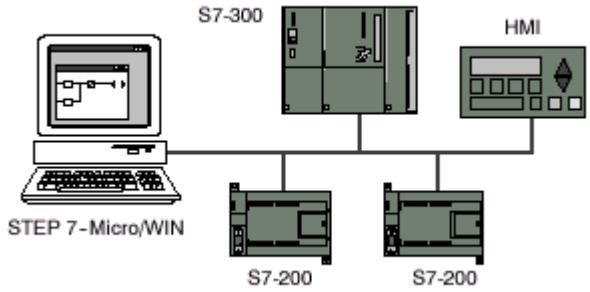
LD V200.7 // Khi bit NETR Done set

AW= VW208 +100 // và đã đóng gói 100 hộp,

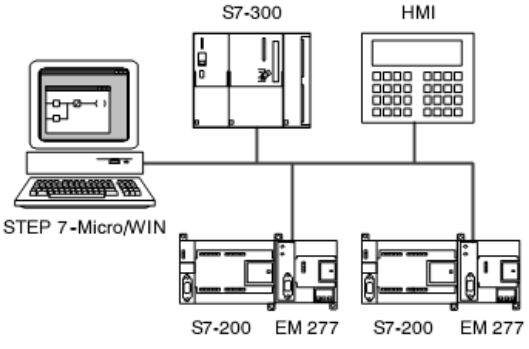
```
MOVB 2 VB301 // Nạp địa chỉ trạm 2
MOVD &VB101 VD302 // Nạp con trỏ dữ liệu trạm 2
MOVB 2 VB306 // Nạp số byte truyền
MOVW +0 VW307 // Nạp dữ liệu truyền
NETW VB300 0 // Xóa số hộp trạm 2 đã đóng gói

NETWORK 3
LD V200.7 // Khi bit NETR Done set ,
MOVB VB207 VB400 // cất dữ liệu điều khiển từ trạm 2
NETWORK 4
LDN SM0.1 // Nếu không phải chu kỳ đầu
AN V200.6 // và không có lỗi ,
AN V200.5
MOVB 2 VB201 // Nạp địa chỉ trạm 2
MOVD &VB100 VD202 // Nạp con trỏ dữ liệu trạm 2
MOVB 3 VB206 // Nạp số byte dữ liệu nhận
NETR VB200 0 // Đọc dữ liệu điều khiển và trạng thái trong trạm 2
```

Ngoài lệnh NETR/W S7-200 có thể giao tiếp mạng dùng chuẩn MODBUS và giao tiếp biến tần dùng chuẩn USS, các giao thức này sử dụng thư viện Modbus Protocol và USS Protocol

S7-200 ở chế độ PPI Slave có thể kết nối với S7-200, S7-300, S7-300 dùng lệnh X_PUT (SFC68), X_GET (SFC67) giao tiếp với S7-200. Cách kết nối này dùng cho vận tốc truyền đến 187.5Kb/s, muốn tăng vận tốc truyền đến 12Mb/s theo mạng Profibus ta dùng module mở rộng EM277 cho S7-200 và card CP cho máy tính, S7-300 phải có cổng giao tiếp Profibus, nếu trong cấu hình phần cứng của Simatic Manager không có EM277 ta download và install GSD (Generic Station Descriptor) của EM277 (file siem089d.gsd)



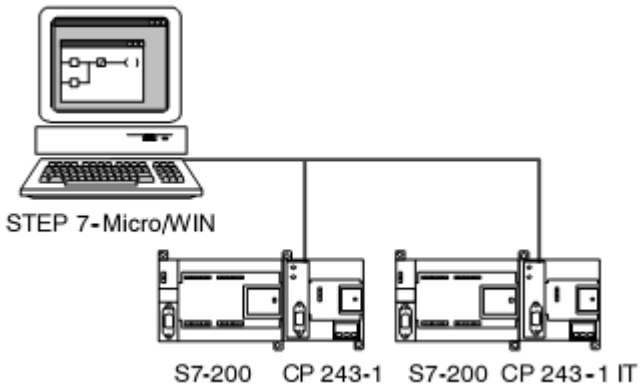
Giao tiếp mạng MPI



Giao tiếp mạng Profibus

Trường hợp mạng LAN 10Mb/100Mb, ta sử dụng module CP243-1 và dùng Ethernet wizard để cài đặt. Muốn kết nối Internet, dùng CP243-IT.

Với những yêu cầu kết nối đơn giản , không yêu cầu vận tốc lớn và khoảng cách xa, ta có thể sử dụng modem có dây và không dây, có thể sử dụng module mở rộng modem EM241 kết nối với CPU hay dùng modem ngoài kết nối qua cổng nối tiếp

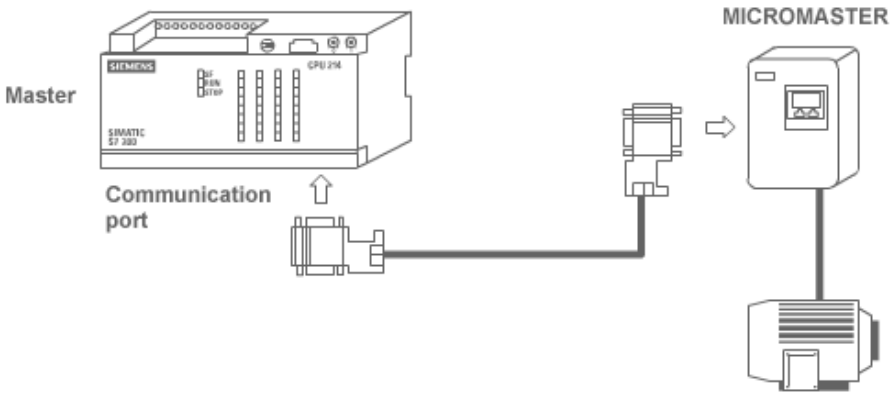


Giao tiếp TCP/IP

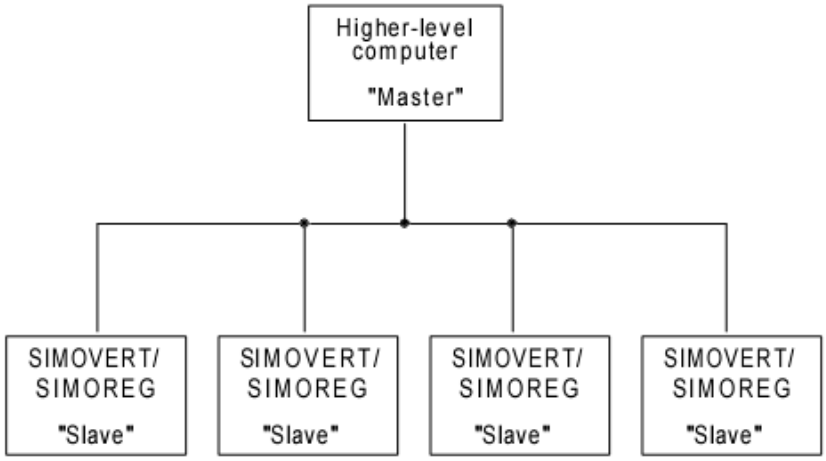
Thư viện USS Protocol

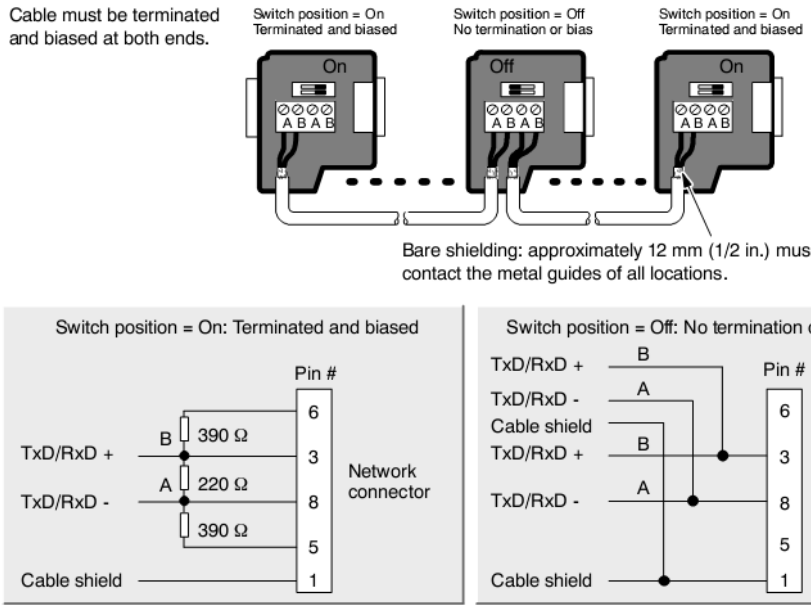
Một PLC S7-200 có thể kết nối nhiều biến tần Micromaster theo mode USS (Universal Serial interface) , việc cài đặt thông số cho biến tần thực hiện nhờ thư viện USS Protocol, giúp các biến tần làm việc đồng bộ với nhau. Một PLC master có thể ghép nối với 31 biến tần slave để điều khiển biến tần và cài đặt thông số.

Mỗi biến tần có địa chỉ cụ thể, ghép với PLC và các biến tần khác theo mạng 485, ta nên dùng PLC có 2 cổng truyền thông (224XP, 226), một cổng ghép với biến tần và cổng thứ hai kết nối máy tính để dễ dàng monitor hoạt động hệ thống dùng MicroWin



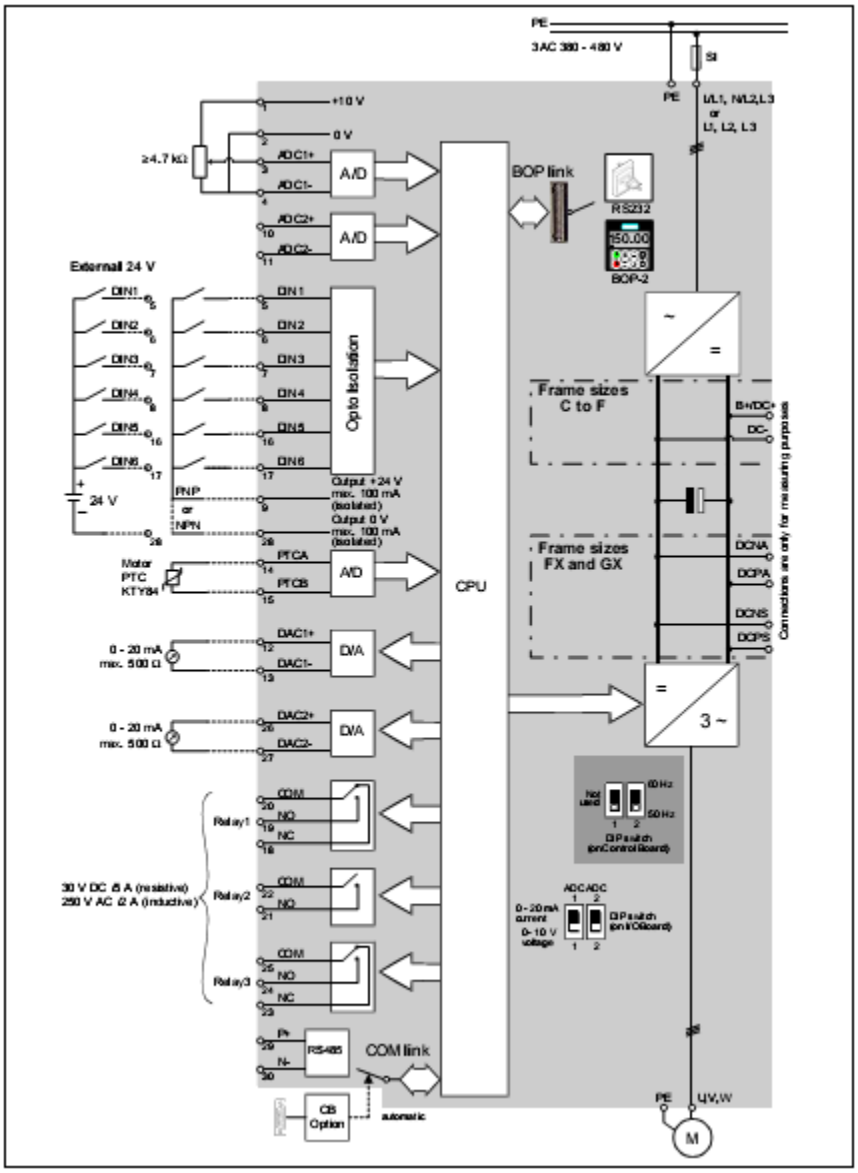
Cáp kết nối các thiết bị với nhau là loại hai dây có vỏ bọc, đầu và cuối cáp mạng có điện trở kết thúc





Biến tần Micromaster họ MM420/ 430/ 440 điều khiển vận tốc động cơ không đồng bộ công suất từ 0,12KW đến 250KW, cấp điện áp 110V 1 pha, 220V 1 pha hay 3 pha 220/380V, 380/460V, tùy theo serie, có thể điều khiển vận tốc động cơ theo kiểu V/f hằng số, V^2/f hằng số, điều khiển dòng từ thông FCC, điều khiển vector, điều khiển moment, điều khiển vòng hở hay vòng kín. Biến tần có ngõ vào analog để điều khiển vận tốc, ngõ vào ra số, cổng truyền thông USS, màn hình và bàn phím lập trình BOP (basic operator panel) hay màn hình trạng thái SDP (status display panel), chi tiết đề nghị xem catalogue của thiết bị.

Đầu tiên ta phải cài đặt thông số cho biến tần thông qua BOP .BOP hiển thị 5 chữ số và có 8 nút nhấn. Các bước cụ thể tùy thuộc loại biến tần

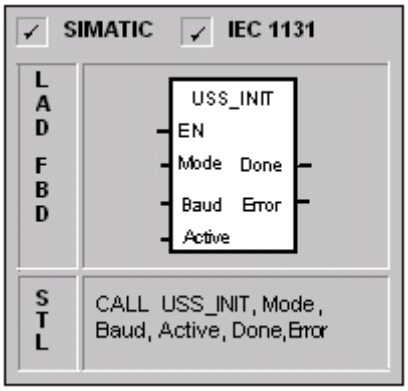


Sơ đồ Biến tần MM430

Các bước lập trình USS Protocol

USS cài đặt 14 chương trình con và 3 chương trình ngắt, dung lượng nhớ chương trình đến 3600 byte, vùng nhớ V chiếm 400 byte, bộ đệm truyền thông 16 byte, dùng bốn ACC.

- Dùng lệnh USS_Init đặt cấu hình truyền thông, lệnh này thực hiện ở một chu kỳ quét bằng cách dùng kích cạnh ở EN,



Mode=1 :port 0 USS

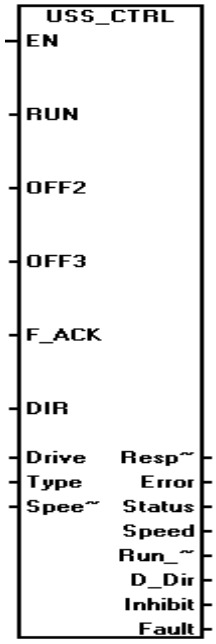
Mode=0 : port 0 PPI

Baud : vận tốc truyền

Active : số nhị phận 32 bit, bit i mức 1 nghĩa là biến tần thứ i được cho phép

Done : 1 cho biết lệnh thực hiện không lỗi

- Lệnh USS_CONTROL điều khiển biến tần



RUN=1 biến tần chạy , Fault, Inhibit, OFF2, OFF3 phải là 0

OFF2=1 : biến tần dừng chậm dần

OFF3=1 : biến tần dừng nhanh

DIR =1: quay thuận

Drive : số từ 0 đến 31 cho biết điều khiển biến tần nào

Type = 0 : biến tần MM3

Type = 1 : biến tần MM4

Speed_SP : đặt vận tốc, số thực phần trăm vận tốc tối đa

Response_R: báo đã nhận trả lời từ biến tần

Speed: vận tốc thực của đông cơ, phần trăm

D_Dir : hướng quay thực của động cơ

Run_En : cho biết động cơ đang chạy hay ngừng

F_Ack : ghi nhận lỗi từ biến tần

Error : một byte cho biết kết quả truyền thông

Status : hai byte trạng thái do biến tần gởi về

Fault: 1 báo có lỗi trong biến tần, muốn xóa bit này phải kha71c phục lỗi và cho F_Ack lên 1

Inhibit: cho biết trạng thái bit inhibit trong biến tần, mức 1 là biến tần bị cấm chạy, muốn xóa bit này, Run, Off2, Off3, Fault phải Off

- Lệnh USS_RPM_X đọc thông số biến tần

- USS_WPM_X đổi thông số biến tần

Ví dụ:

```
NETWORK 1        // Initialize USS Protocol:On the first scan, enable the USS protocol
// for port 0 at 19200 with drive address "0" active
LD    SM0.1
CALL  USS_INIT, 1, 19200, 16#00000001, Q0.0, VB1
NETWORK 2        // Control parameters for Drive 0
LD    SM0.0
CALL  USS_CTRL, I0.0, I0.1, I0.2, I0.3, I0.4, 0, 1, 100.0, M0.0, VB2, VW4, VD6, Q0.1, Q0.2, Q0.3, Q0.4
NETWORK 3        // Read a Word parameter from Drive 0
// Read parameter 5 index 0
LD    I0.5  // Save the state of I0.5 to a temporary
=     L60.0  // L location so that this network will display in LAD.
LD    I0.5  // Save the rising edge pulse of I0.5
EU
=     L63.7  // to a temporary L location so that it can be passed to the subroutine.
LD    L60.0
CALL  USS_RPM_W, L63.7, 0, 5, 0, &VB20, M0.1, VB10, VW12
NETWORK 4        // Write a Word parameter to Drive 0
// Write parameter 2000 index 0
LD    I0.6
=     L60.0
```

```
LD    I0.6
EU
=     L63.7
LDN   SM0.0
=     L63.6
LD    L60.0
CALL  USS_WPM_R, L63.7, L63.6, 0, 2000, 0, 50.0, &VB40, M0.2, VB14
```

9.29 PC ACCESS

PC Access đùng dể giao tiếp S7-200 với các chương trình Visual Basic hay Excel trên máy tính. PLC được cài đặt theo mode PPI slave (hay qua modem, mạng Profibus, Ethernet), một máy tính có thể ghép nhiều PLC với các địa chỉ khác nhau, chương trình PC Acess thu thập thông tin từ các PLC slave và gởi dữ liệu đến chúng, các chương trình ứng dụng như VB, Excel, WinCC sẽ tìm thông tin từ hay gởi thông tin đến PLC thông qua chương trình PCAcess. Cơ chế này goi là OPC (Object Linking and Embedding for Process Control, đối tượng nhúng, kết nối để điều khiển quá trình), OPC dựa trên các công nghệ phần mềm của Windows như OLE, COM (Component Object Model), DCOM(Distributed Component Object Model).

Chuẩn OPC được cung cấp bởi OPC Foundation, khi các nhà sản xuất phần mềm và PLC tuân theo chuẩn này thì một phần mềm sẽ dễ dàng kết nối với PLC. Các thành phần của OPC gồm có :

- OPC-DA (Data Access) : dùng để truy cập dữ liệu thời gian thực của thiết bị chấp hành

- OPC-HDA (Historical Data Access) : dùng để truy cập dữ liệu quá khứ của thiết bị chấp hành để phân tích hoạt động của hệ thống .

- OPC-AE (Alarms & Events) : dùng để giám sát những cảnh báo và sự kiện của quá trình sản xuất .

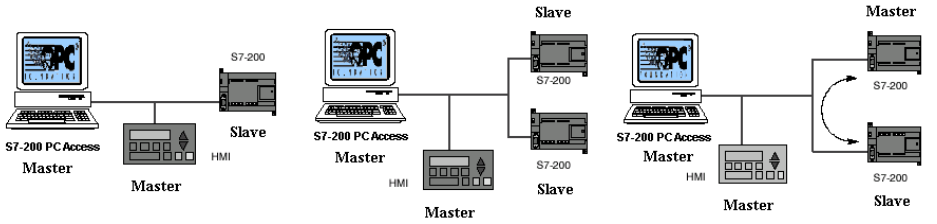
- OPC-DX (Data eXchange) : định nghĩa cách một OPC Server trao đổi thông tin với một OPC Server khác .

- OPC-XML (XML Data Access) : định nghĩa phương thức và định dạng dữ liệu dựa vào chuẩn XML (Extensible Makup Language) cho phép trao đổi và xử lý dữ liệu giữa các hệ điều hành khác nhau như : Windows , Unix , Solaris , …

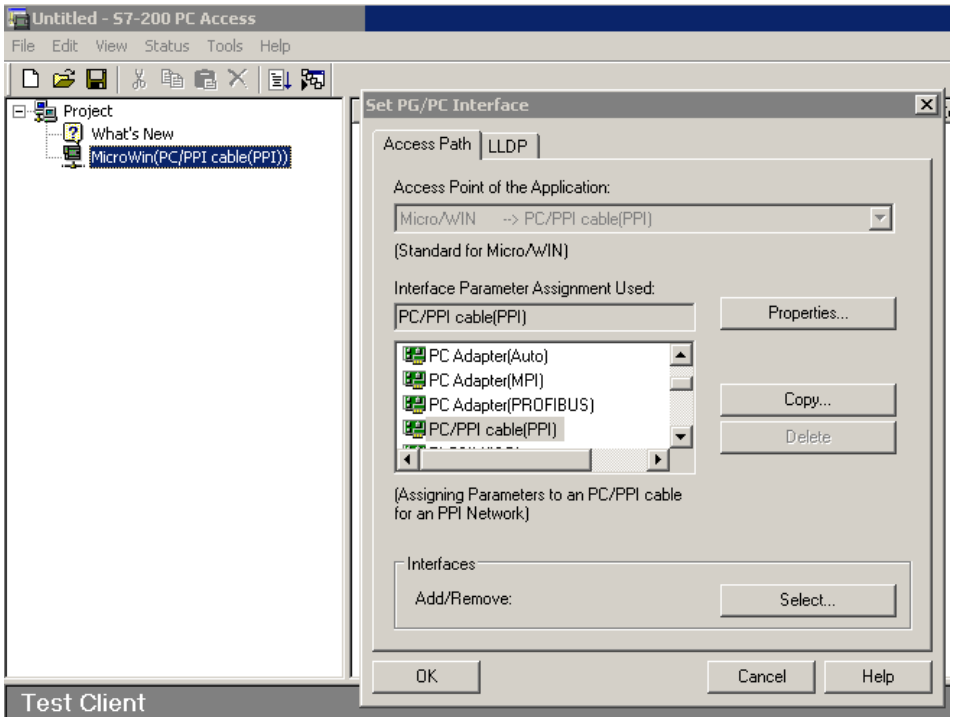
Ngoài ra còn các chuẩn khác như : OPC Security, OPC Batch, OPC Commands, OPC for ERP.

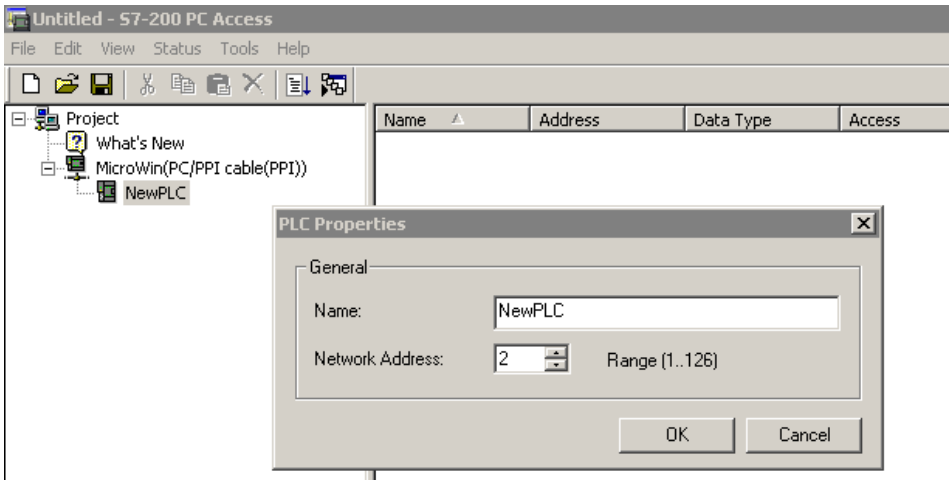
PC Access thuộc loại OPC - DA được gọi là OPC Server và các chương trình ứng dụng gọi là OPC Client (VB, Excel) . PC Access truy cập các địa chỉ trên PLC gọi là item, Client truy cập các item này trên Server. Các item cùng đặc tính có thể gom thành group để dễ quản lý.

Khi sử dụng PC Access ta cài đặt chương trình ứng dụng cho S7-200, nếu kết nối PLC với máy tính qua cáp PPI thì đặt mode PPI slave địa chỉ 2, nếu dùng nhiều PLC thì gán các địa chỉ khác nhau cho PLC (tối đa 4 PLC).

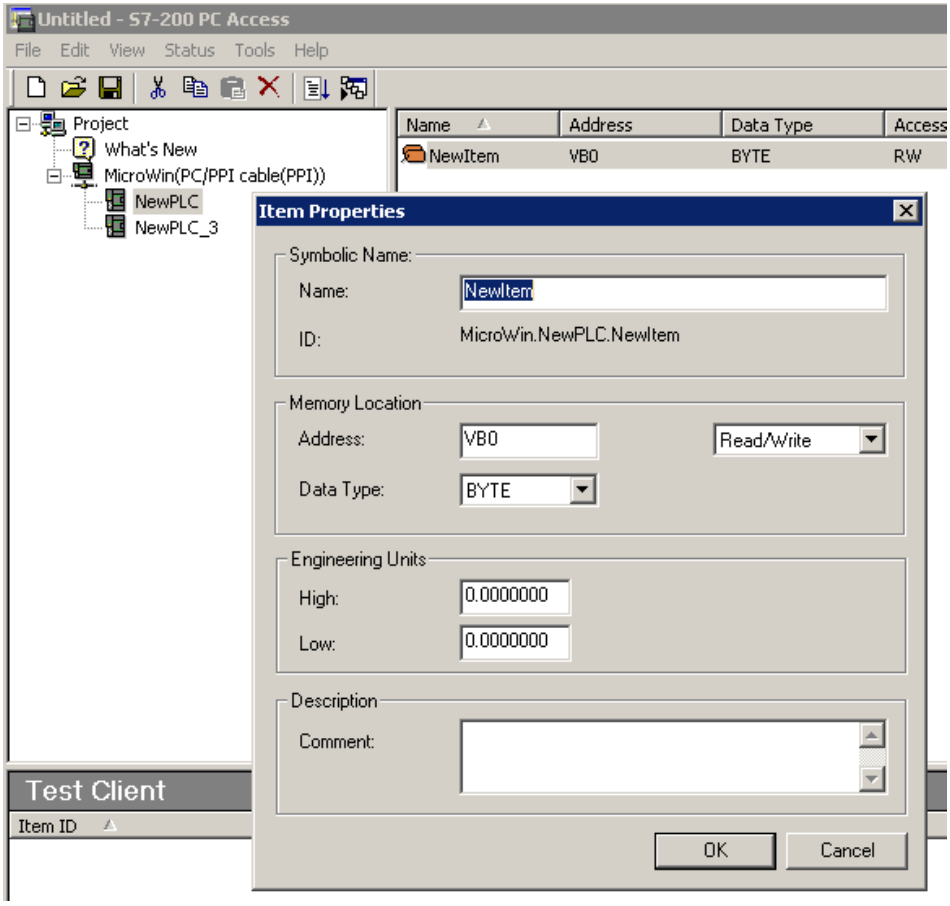


Mở chương trình PC Access, khai báo loại truyền thông (modem và các địa chỉ PLC muốn giao tiếp

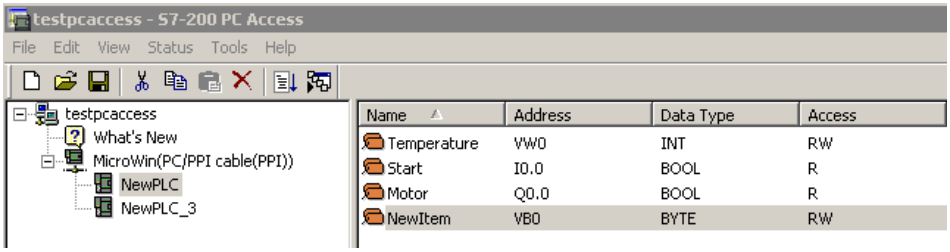




Có thể đặt tên PLC tùy ý, bấm chuột phải vào NewPLC chọn New-Folder hay Item, chọn tên Item, địa chỉ ô nhớ PLC, loại dữ liệu, đọc/ghi, giá trị mức ngưỡng, …



Sau khi hoàn tất, lưu file dưới tên có đuôi pca



Nếu có sẵn PLC có thể kết nối PLC với máy tính, chép các item vào cửa sổ Test Client, bấm Status- Start Test Client, ta sẽ thấy các dữ liệu từ PLC được đưa lên máy tính

Test Client

Item ID	Value	Time Stamp	Quality
MicroWin.NewPLC.Temperature	+00	22:10:25:796	Good
MicroWin.NewPLC.Start	0	22:10:25:796	Good
MicroWin.NewPLC.NewItem	00	22:10:25:796	Good
MicroWin.NewPLC.Motor	0	22:10:25:796	Good

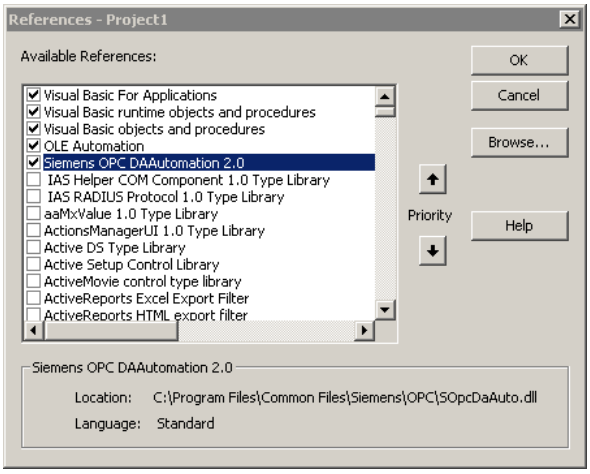
Mỗi item được gán Item ID và Run Time Name để các client truy cập

Name	Item ID	Run-Time Name
Motor	MicroWin.NewPLC.Motor	MicroWin:2:0.0.0.0:0000:0000,Q0.0,BOOL,R,0.0000000,0.0000000
NewItem	MicroWin.NewPLC.NewItem	MicroWin:2:0.0.0.0:0000:0000,VB0,BYTE,RW,0.0000000,0.0000000
Start	MicroWin.NewPLC.Start	MicroWin:2:0.0.0.0:0000:0000,I0.0,BOOL,R,0.0000000,0.0000000
Temperature	MicroWin.NewPLC.Temperature	MicroWin:2:0.0.0.0:0000:0000,VW0,INT,RW,1000.000,200.0000

Chương trình client có thể tạo thêm các item mới không có sẵn trên server bằng Run Time Name, các biến ký hiệu có sẵn trong chương trình của PLC cũng có thể import vào server.

Sau khi đã kiểm tra việc kết nối, ta lưu file rồi đóng PC Access, các chương trình client sẽ kết nối tự động với project được lưu cuối cùng trong PC Access, lúc đó trên system tray của Windows xuất hiện icon Simatic S7-200 PC Access OPC Server.

Sau khi đã cài đặt PC Access, ta viết chương trình Visual Basic. VB dùng các hàm trong thư viện Siemens OPC DAAutomation thư mục SOpc DaAuto.dll , ta phải khai báo thư viện này bằng cách vào Project-References- check Siemens OPC DAAutomation- OK.



Sau đó khai báo các biến:

Private MyOPCServer As OPCServer 'OPCServer Object

Private MyGroups As OPCGroups 'OPCGroups Collection Object

Private WithEvents MyGroup As OPCGroup 'OPCGroup Object

Private MyItems As OPCItems ' OPCItems Collection Object

Private MyItemServerHandles() As Long 'Server Handles for Items

Dim MyTID As Long ' Transaction ID for asynchronous calls

Visual Basic Client kết nối với S7-200 PC Access Server dùng method:

- Set MyOPCServer = New OPCServer
- MyOPCServer.Connect (“S7200.OPCserver”)

Cắt kết nối dùng lệnh

- MyOPCServer.DisConnect
- Set MyOPCServer = Nothing

Để tạo ra Group và Items thì gọi method:

- Set MyGroups = MyOPCServer.OPCGroups
- Set MyGroup =MyGroups.Add (“AnOPCGroupName”)
- MyOPCItemCollection.AddItems (AddItemCount, MyOPCItemIDs, ClientHandles,

MyOPCItemServerHandles,
MyOPCItemServerError,
MyOPCRequestedDataType, MyOPCAccessPaths)

Lúc này, Automation Visual Basic Client có thể tiến hành truy cập dữ liệu từ S7-200 PC Access server.

Đọc các Value, Quality, Time Stamp của các Item:

- OneGroup.SyncRead (Source, NumItems, ServerHandles, Values, Errors, Qualities, TimeStamps)
- OneGroup.AsyncRead (NumItems, ServerHandles, Errors, ClientTransactionID, ServerTransactionID)

Sử dụng method để viết các Value đến các Item :

- OneGroup.SyncWrite (NumItems, ServerHandles, Values, Errors)
- OneGroup.AsyncWrite (NumItems, ServerHandles, Values,Errors, ClientTransactionID, ServerTransactionID)

Gỡ bỏ các Group hay Item:

- MyGroups.Remove (“AnOPCGroupName”)
- MyGroups.RemoveAll
- AnOPCItemCollection.Remove (AnOPCItemServerHandles, AnOPCItemServerErrors)

Để cắt kết nối Automation Visual Basic Client với S7-200 PC Access (OPC Server) thì sử dụng method :

- AnOPCServer.Disconnect

Các method ở trên được miêu tả chi tiết trong “ Data Access Automation Standard” của OPC Foundation

Kết nối Excel

S7-200 PC Access đưa Visual Basic For Appliciation (VBA) vào trong Microsoft Excel cho phép Microsoft Excel có thể thu thập

dữ liệu từ OPC Server. VBA làm việc thông qua OPC Automation Wrapper để Excel có thể truy cập đến S7-200 PC Access Server. Việc truy cập bằng Excel giúp lưu trữ được dữ liệu

Để có thể kết nối Excel với S7-200 PC Access, phải đưa "OPCS7-200ExcelAddin.xla" của S7-200 PC Access vào Excel bằng cách mở Tool-Add_Ins tìm thư mục cài đặt S7 200 PC Access\bin chọn OPCS7200ExcelAddin . Lúc này trong cửa sổ của Microsoft Excel xuất hiện bốn Button:

- Formula-Wizard: cho phép duyệt và lựa chọn Items muốn đọc đã có sẵn trong PC Access.

- Write-Wizard: cho phép duyệt và lựa chọn Items muốn viết giá trị mới vào, các Item đã có sẵn trong PC Access.

- Start collecting data: bắt đầu giao tiếp với S7-200 PC Access.

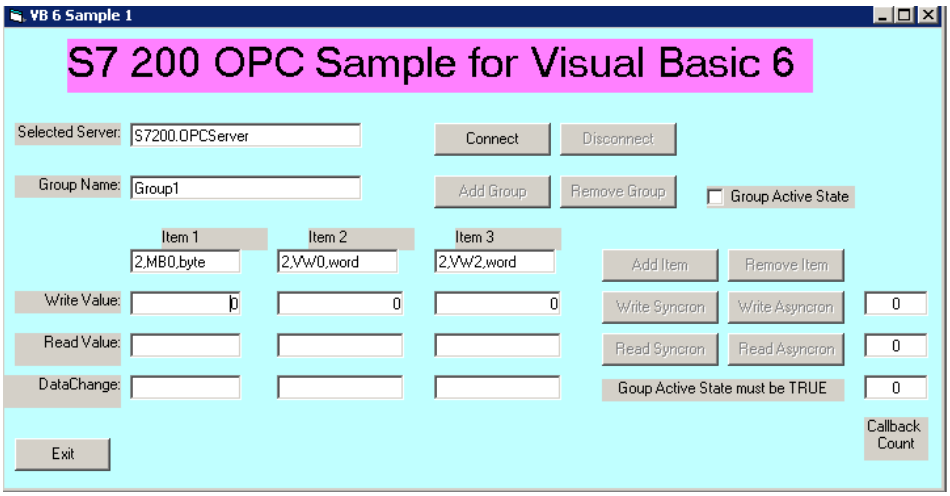
- Stop collecting data: kết thúc giao tiếp với S7-200 PC Access.

Thông qua bốn button này, Excel có thể truy cập được dữ liệu trong PC Access, từ đó có thể vẽ được đồ thị của dữ liệu và có thể lưu trữ dữ liệu.

Thông qua phần mềm S7-200 PC Access, các Automation Client có thể truy cập đến PLC Simatic một cách đơn giản và nhanh chóng thông qua các method ở trên. Trên PLC S7-200, ta không phải dùng các lệnh nhận hay truyền dữ liệu của PLC.

Ví dụ:

Chương trình VB mẫu




```
Option Explicit
Option Base 1   ' All OPC Automation Arrays start with 1
Private MyOPCServer As OPCServer        ' OPCServer Object
Private MyGroups As OPCGroups           ' OPCGroups Collection Object
Private WithEvents MyGroup As OPCGroup  ' OPCGroup Object
Private MyItems As OPCItems             ' OPCItems Collection Object
Private MyItemServerHandles() As Long   ' Server Handles for Items
Dim MyTID As Long                       ' Transaction ID for asynchronous calls

Private Sub cmdConnect_Click()
   On Error GoTo ErrorHandler
   Set MyOPCServer = New OPCServer          ' Create OPCServer Object
   Call MyOPCServer.Connect("S7200.OPCServer")   ' Connect To OPC Server
     ' Set Button Enable
     cmdConnect.Enabled = False
     cmdDisconnect.Enabled = True
     cmdAddGroup.Enabled = True
       Exit Sub
   ErrorHandler:
   MsgBox Err.Description + Chr(13) + "Connecting to OPC Server", vbCritical, "ERROR"
End Sub

Private Sub cmdDisconnect_Click()
   On Error GoTo ErrorHandler
   MyOPCServer.Disconnect          ' Disconnect from OPC Server
   Set MyOPCServer = Nothing       ' Delete OPCServer Object
     ' Set Button Enable
```

```
    cmdDisconnect.Enabled = False
    cmdAddGroup.Enabled = False
    cmdConnect.Enabled = True
    Exit Sub

    ErrorHandler:

    MsgBox Err.Description + Chr(13) + "Disconnecting from OPC Server", vbCritical, "ERROR"

End Sub

Private Sub cmdAddGroup_Click()
    On Error GoTo ErrorHandler


    Set MyGroups = MyOPCServer.OPCGroups        ' Get OPCGroups Collection Object from
MyOPCServer

    ' Set Default Properties for Group Collection
    ' These Properties are used to set the Properies for new Groups

    MyGroups.DefaultGroupUpdateRate  = 500   ' Set Default Group Update Rate to 500 ms

    MyGroups.DefaultGroupIsActive = False ' Set Default Group Active State to Inactive

    Set MyGroup = MyGroups.Add(txtGroup.Text)   ' Add a new Group to the Group Collection

    ' Set Group Properties
      MyGroup.IsSubscribed = True   ' Enable Callbacks
      If CheckGroupActive.Value = 1 Then
      MyGroup.IsActive = True
      Else
      MyGroup.IsActive = False
      End If
      ' Set Button Enable
      cmdAddGroup.Enabled = False
      cmdDisconnect.Enabled = False
      cmdRemGroup.Enabled = True
      cmdAddItem.Enabled = True
    Exit Sub

    ErrorHandler:

    MsgBox Err.Description + Chr(13) + "Adding a Group to OPC Server", vbCritical, "ERROR"

End Sub

Private Sub cmdRemGroup_Click()
      On Error GoTo ErrorHandler
       MyGroups.RemoveAll      ' Removes all Groups

       Set MyGroup = Nothing   ' Delete OPCGroup Object


    Set MyGroups = Nothing  ' Delete OPCGroups Collection Object

      ' Set Button Enable
      cmdRemGroup.Enabled = False
      cmdAddItem.Enabled = False
```

```
    cmdAddGroup.Enabled = True
    cmdDisconnect.Enabled = True
      Exit Sub

      ErrorHandler:

  MsgBox Err.Description + Chr(13) + "Removing Group from OPC Server", vbCritical,
"ERROR"

      End Sub

Private Sub CheckGroupActive_Click()

    If Not MyGroup Is Nothing Then
    If CheckGroupActive.Value = 1 Then
      MyGroup.IsActive = True

  Else

     MyGroup.IsActive = False
    End If
    End If
End Sub

Private Sub cmdAddItem_Click()
    On Error GoTo ErrorHandler
    Dim i As Long
    Dim ErrorFlag As Boolean
    Dim ItemObj As OPCItem
    Dim ItemIDs(3) As String
    Dim ItemClientHandles(3) As Long
  Dim Errors() As Long          ' Array for returned Item related errors

  ErrorFlag = False

  Set MyItems = MyGroup.OPCItems        ' Get OPCItems Collection Object from
MyOPCServer

    ' Initialize the [IN] parameters for the Add Items call
    ' ItemIDs -> ItemIDs of the Items to add
  ' ItemClientHandles -> Client defined handles for the Items. The Server sends these
handles in the Callbacks

      ItemIDs(1) = txtItem1.Text   ' Read ItemId 1 from Text Box
      ItemIDs(2) = txtItem2.Text   ' Read ItemId 2 from Text Box
      ItemIDs(3) = txtItem3.Text   ' Read ItemId 3 from Text Box
     ItemClientHandles(1) = 1
    ItemClientHandles(2) = 2
    ItemClientHandles(3) = 3
    ' [OUT] parameters are
  ' ItemServerHandles -> Server defined handles for the Items. The client must use these
handles for all Read/Write calls

  ' Errors -> Item related errors

     ' Add Items to the Group
  Call MyItems.AddItems(3, ItemIDs, ItemClientHandles, MyItemServerHandles, Errors)
```

```
    ‘ Check Item Errors
    For i = 1 To 3
    If Not Errors(i) = 0 Then
       MsgBox “Item “ + Str$(i) + “ FAILED. Error Code = “ + Str$(Errors(i)), vbCritical

       ErrorFlag = True

    End If
    Next
    ‘ Continue only if all Items SUCCEEDED
    If ErrorFlag Then
    Dim RemoveErrors() As Long
    Dim RemoveHandles(1) As Long
    ‘ Remove Succeede Items
    For i = 1 To 3
       If Errors(i) = 0 Then

       RemoveHandles(1) = MyItemServerHandles(i)

       Call MyItems.Remove(1, RemoveHandles, RemoveErrors)

       End If

    Next
    Else
    ‘ Set Button Enable
    cmdAddItem.Enabled = False
    cmdRemGroup.Enabled = False
    cmdRemItem.Enabled = True
    cmdWriteSync.Enabled = True
    cmdWriteAsync.Enabled = True
    cmdReadSync.Enabled = True
    cmdReadAsync.Enabled = True
    End If
       Exit Sub

       ErrorHandler:

  MsgBox Err.Description + Chr(13) + “Adding Items to the Group”, vbCritical, “ERROR”

End Sub

Private Sub cmdRemItem_Click()
     On Error GoTo ErrorHandler
  Dim i As Long

  Dim Errors() As Long          ‘ Array for returned Item related errors

     ‘ Remove Items from the Group
  Call MyItems.Remove(3, MyItemServerHandles, Errors)

    ‘ Check Item Errors
    For i = 1 To 3
  If Not Errors(i) = 0 Then MsgBox “Item “ + Str$(i) + “ FAILED. Error Code = “ +
Str$(Errors(i)), vbCritical

  Next

  Erase MyItemServerHandles     ‘ Erase Item Server Handle Array
```

```
    ' Set Button Enable
    cmdRemItem.Enabled = False
    cmdWriteSync.Enabled = False
    cmdWriteAsync.Enabled = False
    cmdReadSync.Enabled = False
    cmdReadAsync.Enabled = False
    cmdAddItem.Enabled = True
    cmdRemGroup.Enabled = True
      Exit Sub

      ErrorHandler:

  MsgBox Err.Description + Chr(13) + "Removing Items from the Group", vbCritical, "ERROR"

      End Sub

Private Sub cmdWriteSync_Click()
    On Error GoTo ErrorHandler
    Dim i As Long
    Dim Values(3) As Variant
  Dim Errors() As Long          ' Array for returned Item related errors

    ' Initialize the [IN] parameters for the SyncWrite call
    ' Values -> Values to write

      Values(1) = txtWriteVal1.Text   ' Read Value 1 from Text Box

      Values(2) = txtWriteVal2.Text   ' Read Value 2 from Text Box

      Values(3) = txtWriteVal3.Text   ' Read Value 3 from Text Box
  ' ItemServerHandles -> Server defined handles from the AddItems call

     ' Write Values Syncronous
  Call MyGroup.SyncWrite(3, MyItemServerHandles, Values, Errors)

    ' Check Item Errors
    For i = 1 To 3
  If Not Errors(i) = 0 Then MsgBox "Item " + Str$(i) + " FAILED. Error Code = " +
Str$(Errors(i)), vbCritical

  Next

     Exit Sub
    ErrorHandler:

  MsgBox Err.Description + Chr(13) + "Writing Items Syncronous", vbCritical, "ERROR"

End Sub

Private Sub cmdReadSync_Click()
    On Error GoTo ErrorHandler
    Dim i As Long
    Dim Values() As Variant

      Dim Errors() As Long        ' Array for returned Item related errors

      Dim Qualities As Variant   ' Array for returned Qualities of the Values
  Dim TimeStamps As Variant  ' Array for returned Timestamps of the Values

  ' [IN] parameters for the SyncRead call
```

```
    ' ItemServerHandles -> Server defined handles from the AddItems call

        ' Read Values Syncronous
    Call MyGroup.SyncRead(OPCDevice, 3, MyItemServerHandles, Values, Errors, Qualities,
TimeStamps)

      ' Check [OUT] Parameters
      For i = 1 To 3
      If Not Errors(i) = 0 Then
         MsgBox "Item " + Str$(i) + " FAILED. Error Code = " + Str$(Errors(i)), vbCritical

    Else

      ' Values -> Values from read
      ' Qualities -> Qualities of the returned values
      If Qualities(i) = 192 Then
         txtReadVal.Item(i - 1).Text = Values(i) ' Write Value to Text Box

         txtReadVal.Item(i - 1).BackColor = &HFFFFFF

       Else
         txtReadVal.Item(i - 1).Text = GetQualityText(Qualities(i))

         txtReadVal.Item(i - 1).BackColor = &H8080FF

       End If
      End If
      Next


       Exit Sub
      ErrorHandler:

    MsgBox Err.Description + Chr(13) + "Reading Items Syncronous", vbCritical, "ERROR"

End Sub

Private Sub cmdWriteAsync_Click()
      On Error GoTo ErrorHandler
      Dim i As Long
      Dim Values(3) As Variant
         Dim Errors() As Long          ' Array for returned Item related errors

         Dim CID As Long                    ' CancelID, servergenerierter Wert, mit dem die
Transaktion identifiziert


      ' Initialize the [IN] parameters for the SyncWrite call
      ' Values -> Values to write
         Values(1) = txtWriteVal1.Text   ' Read Value 1 from Text Box

         Values(2) = txtWriteVal2.Text   ' Read Value 2 from Text Box

         Values(3) = txtWriteVal3.Text   ' Read Value 3 from Text Box
    ' ItemServerHandles -> Server defined handles from the AddItems call

    MyTID = MyTID + 1   ' Increment Transaction ID
```

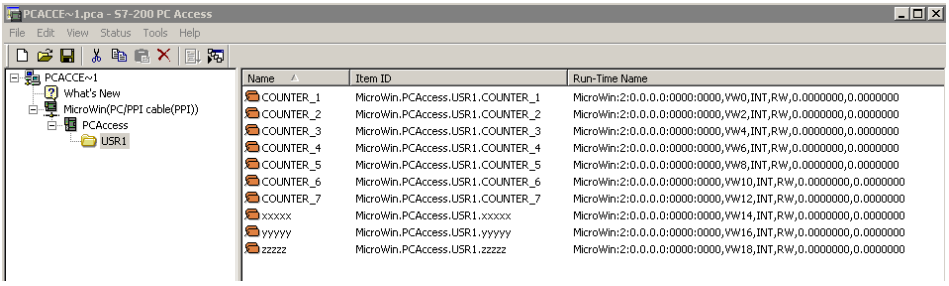
```
        ' Write Values Asyncronous
    Call MyGroup.AsyncWrite(3, MyItemServerHandles, Values, Errors, MyTID, CID)

      ' Check Item Errors
      For i = 1 To 3
    If Not Errors(i) = 0 Then MsgBox "Item " + Str$(i) + " FAILED. Error Code = " +
Str$(Errors(i)), vbCritical

    Next

       Exit Sub
      ErrorHandler:

    MsgBox Err.Description + Chr(13) + "Writing Items Asyncronous", vbCritical, "ERROR"

End Sub

Private Sub cmdReadAsync_Click()
       On Error GoTo ErrorHandler
    Dim i As Long

          Dim Errors() As Long        ' Array for returned Item related errors

          Dim CID As Long                    ' CancelID, servergenerierter Wert, mit dem die
Transaktion identifiziert
    ' [IN] parameters for the AsyncRead call

    ' ItemServerHandles -> Server defined handles from the AddItems call

    MyTID = MyTID + 1   ' Increment Transaction ID

        ' Read Values Syncronous
    Call MyGroup.AsyncRead(3, MyItemServerHandles, Errors, MyTID, CID)

      ' Check Item Errors
      For i = 1 To 3
    If Not Errors(i) = 0 Then MsgBox "Item " + Str$(i) + " FAILED. Error Code = " +
Str$(Errors(i)), vbCritical

    Next

       Exit Sub
      ErrorHandler:

    MsgBox Err.Description + Chr(13) + "Reading Items Asyncronous", vbCritical, "ERROR"

End Sub

        ' Callback from AsyncRead
Private Sub MyGroup_AsyncReadComplete(ByVal TransactionID As Long, ByVal NumItems
As Long, ClientHandles() As Long, ItemValues() As Variant, Qualities() As Long,
TimeStamps() As Date, Errors() As Long)

       On Error GoTo ErrorHandler
    Dim i As Long

       TxtAReadComplete.Text = TxtAReadComplete.Text + 1
      ' Check Parameters
      For i = 1 To NumItems
      If Not Errors(i) = 0 Then
```

```
    MsgBox “AsyncReadComplete Item Clienthandle = “ + Str$(ClientHandles(i)) + “ FAILED.
Error Code = “ + Str$(Errors(i)), vbCritical

        ElseIf ClientHandles(i) > 0 And ClientHandles(i) < 4 Then
      ‘ Values -> Values from read complete
      ‘ Qualities -> Qualities of the values
      If Qualities(i) = 192 Then
   txtReadVal.Item(ClientHandles(i) - 1).Text = ItemValues(i) ‘ Write Value to Text Box

          txtReadVal.Item(ClientHandles(i) - 1).BackColor = &HFFFFFF

        Else
          txtReadVal.Item(ClientHandles(i) - 1).Text = GetQualityText(Qualities(i))

          txtReadVal.Item(ClientHandles(i) - 1).BackColor = &H8080FF

        End If
   Else

   MsgBox “AsyncWriteComplete Item “ + Str$(i) + “ has invalid Client Handle “, vbCritical

     End If
     Next
        Exit Sub

        ErrorHandler:

   MsgBox Err.Description + Chr(13) + “Async Read Complete”, vbCritical, “ERROR”

End Sub

        ‘ Callback from AsyncWrite
Private Sub MyGroup_AsyncWriteComplete(ByVal TransactionID As Long, ByVal NumItems
As Long, ClientHandles() As Long, Errors() As Long)

        On Error GoTo ErrorHandler
   Dim i As Long

        TxtAWriteComplete.Text = TxtAWriteComplete.Text + 1
      ‘ Check Item Errors
      For i = 1 To NumItems
   If Not Errors(i) = 0 Then MsgBox “AsyncWriteComplete Item Clienthandle = “ +
Str$(ClientHandles(i)) + “ FAILED. Error Code = “ + Str$(Errors(i)), vbCritical

        Next
          Exit Sub

          ErrorHandler:

   MsgBox Err.Description + Chr(13) + “Async Write Complete”, vbCritical, “ERROR”

End Sub

        ‘ Callback from OnDataChange
Private Sub MyGroup_DataChange(ByVal TransactionID As Long, ByVal NumItems As Long,
ClientHandles() As Long, ItemValues() As Variant, Qualities() As Long, TimeStamps() As
Date)

   On Error GoTo ErrorHandler
   Dim i As Long
```

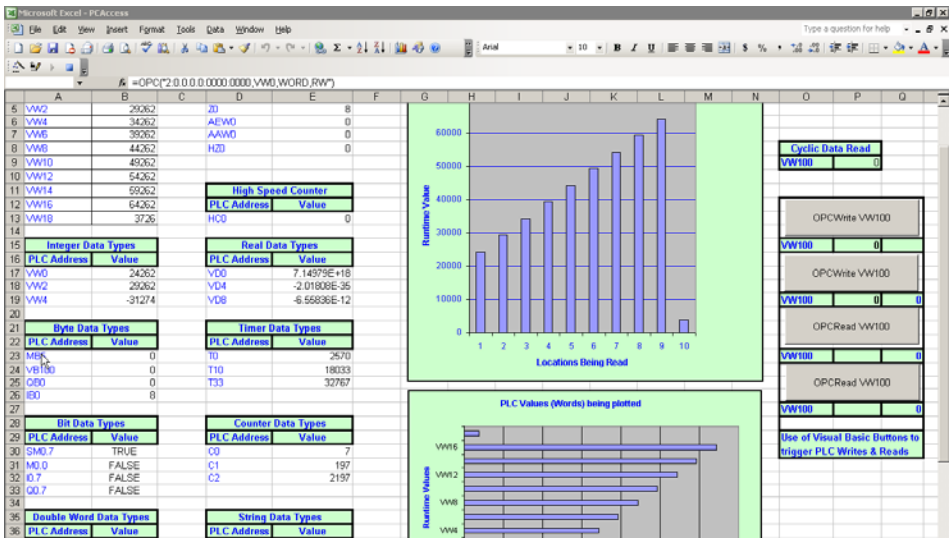
```
    TxtDataChange.Text = TxtDataChange.Text + 1
    ‘ Check Parameters
      For i = 1 To NumItems
      If ClientHandles(i) > 0 And ClientHandles(i) < 4 Then
            ‘ Values -> Values from read complete
            ‘ Qualities -> Qualities of the values
            If Qualities(i) = 192 Then
            txtChangeVal.Item(ClientHandles(i) - 1).Text = ItemValues(i) ‘ Write Value to Text
Box
            txtChangeVal.Item(ClientHandles(i) - 1).BackColor = &HFFFFFF
        Else
          txtChangeVal.Item(ClientHandles(i) - 1).Text = GetQualityText(Qualities(i))
          txtChangeVal.Item(ClientHandles(i) - 1).BackColor = &H8080FF
        End If
    Else
          MsgBox “DataChange Item “ + Str$(i) + “ has invalid Client Handle “, vbCritical
      End If
      Next
          Exit Sub
          ErrorHandler:
    MsgBox Err.Description + Chr(13) + “OnDataChange”, vbCritical, “ERROR”
End Sub
Private Sub cmdExit_Click()
        Unload Me
End Sub
      ‘ Load Form Event
Private Sub Form_Load()
    MyTID = 1          ‘ Reset Transaction ID
End Sub
‘ Unload Form Event
Private Sub Form_Unload(Cancel As Integer)
    If cmdRemItem.Enabled = True Then Call cmdRemItem_Click
    If cmdRemGroup.Enabled = True Then Call cmdRemGroup_Click
    If cmdDisconnect.Enabled = True Then Call cmdDisconnect_Click
End Sub
Private Function GetQualityText(Quality) As String
      Select Case Quality
            Case 0:     GetQualityText = “BAD”
            Case 64:    GetQualityText = “UNCERTAIN”
```

```
        Case 192:   GetQualityText = "GOOD"
        Case 8:     GetQualityText = "NOT_CONNECTED"
        Case 13:    GetQualityText = "DEVICE_FAILURE"
        Case 16:    GetQualityText = "SENSOR_FAILURE"
        Case 20:    GetQualityText = "LAST_KNOWN"
        Case 24:    GetQualityText = "COMM_FAILURE"
        Case 28:    GetQualityText = "OUT_OF_SERVICE"
        Case 132:   GetQualityText = "LAST_USABLE"
        Case 144:   GetQualityText = "SENSOR_CAL"
        Case 148:   GetQualityText = "EGU_EXCEEDED"
        Case 152:   GetQualityText = "SUB_NORMAL"
        Case 216:   GetQualityText = "LOCAL_OVERRIDE"
        Case Else: GetQualityText = "UNKNOWN QUALITY"
    End Select
End Function
```

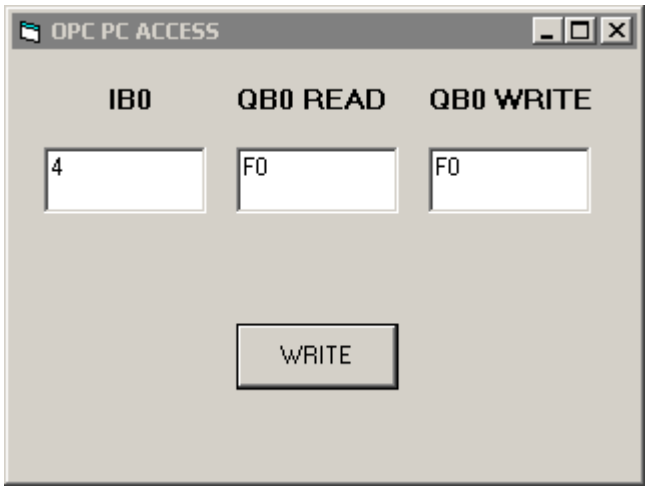
Chương trình PC Access mẫu



Chương trình Excel mẫu



Dựa trên chương trình mẫu ta viết một chương trình VB đơn giản đọc IB0, QB0 theo chu kỳ 1sec và ghi vào QB0, trong PLC cài đặt cổng truyền thông PPI Slave




```
Option Explicit
Option Base 1 ' All OPC Automation Arrays start with 1
Private Server As OPCServer        ' OPCServer Object
Private NhomPLC1 As OPCGroups          ' OPCGroups Collection Object
Private WithEvents Nhom As OPCGroup  ' OPCGroup Object
Private DataPLC As OPCItems            ' OPCItems Collection Object
Private ItemHandles() As Long   ' Server Handles for Items


Private Sub Command1_Click()
```

```
    Dim Values(2) As Variant
    Dim Errors() As Long          ' Array for returned Item related errors
    Values(2) = "&H" & Text3.Text   ' Read Value 2 from Text3 Box
    Call Nhom.SyncWrite(2, ItemHandles, Values, Errors)
End Sub

Private Sub Form_Load()
    Dim ItemIDs(2) As String
    Dim ItemClientHandles(2) As Long
    Dim ItemObj As OPCItem
    Dim Errors() As Long
    Set Server = New OPCServer          ' Create OPCServer Object
    Call Server.Connect("S7200.OPCServer")  ' Connect To OPC Server
    Set NhomPLC1 = Server.OPCGroups
    NhomPLC1.DefaultGroupUpdateRate = 500
    NhomPLC1.DefaultGroupIsActive = True
    Set Nhom = NhomPLC1.Add    ' Add a new Group to the Group Collection
    Nhom.IsSubscribed = True   ' Enable Callbacks
    Nhom.IsActive = True
    Set DataPLC = Nhom.OPCItems
    ItemIDs(1) = "2,IB0,byte"              '2: PLC Address
    ItemIDs(2) = "2,QB0,byte"
    ItemClientHandles(1) = 1
    ItemClientHandles(2) = 2
    Call DataPLC.AddItems(2, ItemIDs, ItemClientHandles, ItemHandles, Errors)
    '2: number of items
End Sub

Private Sub Timer1_Timer()
    Dim Errors() As Long
    Dim Values() As Variant
    Call Nhom.SyncRead(OPCDevice, 2, ItemHandles, Values, Errors)
    Text1.Text = Hex(Values(1))
    Text2.Text = Hex(Values(2))
End Sub
```

Khi chương trình chạy, Simatic s7-200 OPC Server sẽ được gọi truyền thông với PLC.

Ta có thể dùng nhiều OPC Server khác để kết nối Visual Basic với S7-200 ví dụ như KEPServerEX của Kepware http://www.kepware.com/Products/visual-basic.asp

Một cách khác là dùng phần mềm SCADA

9.30 MODBUS

Có hai loại truyền thông Modbus ASCII và Modbus RTU. Trong Modbus ASCII mỗi nửa byte nhị phân đổi thành byte mã ASCII của số từ 0 đến F, cấu trúc bản tin như sau

START	ADDRESS	FUNCTION	DATA	LRC CHECK	END
1 CHAR :	2 CHARS	2 CHARS	*n* CHARS	2 CHARS	2 CHARS CRLF

Byte START là dấu : mã 3A, hai byte kết thúc CRLF mã 0D0A, sau START lần lượt là địa chỉ trạm slave, mã hàm, dữ liệu và kiểm tra lỗi LRC longitudinal redundancy check. LRC là kết quả cộng các byte bản tin từ ADDRESS đến DATA không lấy số nhớ rồi bù hai kết quả . Mỗi byte truyền đi gồm có bit start, 7 bit dữ liệu , bit parity , một bit stop (hai bit nếu không dùng parity).

Mode ASCII cho phép khỏang nghỉ giữa hai ký tự của bản tin lên đến 1 sec

Modbus RTU truyền mỗi byte nguyên dạng theo frame sau:

START	ADDRESS	FUNCTION	DATA	CRC CHECK	END
T1–T2–T3–T4	8 BITS	8 BITS	*n* x 8 BITS	16 BITS	T1–T2–T3–T4

Bản tin bắt đầu sau khoảng nghỉ ít nhất 3,5 thời gian một byte, byte đầu tiên là địa chỉ rồi đến mả hàm, dữ liệu, mã sửa sai CRC cyclic redundancy check, sau đó là khoảng nghỉ ít nhất 3,5 thời gian một byte. Cách tạo CRC phức tạp, đầu tiên thanh ghi 16 bit được nạp các bit 1, byte 8 bit đầu tiên trong bản tin exclusive or với thanh ghi , kết quả được dời phải, bit MSB thay bằng 0, nếu LSB đời ra là 1 thanh ghi được exclusive or với một giá trị định trước A001, nếu là 0 thì không làm gì, sau khi dời 8 lần byte kế của bàn tin lại được tiếp tục exor như trên ; sau cùng CRC là hai byte, byte thấp được truyền trước, byte cao truyền sau.

Địa chỉ là của slave gồm hai ký tự (ASCII) hay 8 bit (RTU)

đi từ 1 đến 247 (01 .. F7), khi master dùng địa chỉ 0 tức là bản tin quảng bá dùng cho mọi slave.

Mả hàm được gởi xuống slave

01Read Coil Status	13 Program Controller
02 Read Input Status	14 Poll Controller
03Read Holding Registers	15 Force Multiple Coils
04 Read Input Registers	16 Preset Multiple Registers
05Force Single Coil	17 Report Slave ID
06 Preset Single Register	18 Program 884/M84
07 Read Exception Status	19 Reset Comm. Link
08 Diagnostics (see Chapter 3)	20 Read General Reference
09 Program 484	21 Write General Reference
10 Poll 484	22 Mask Write 4X Register
11 Fetch Comm. Event Ctr.	23 Read/Write 4X Register
12 Fetch Comm. Event Log	24 Read FIFO Queue

S7-200 hỗ trợ các hàm sau

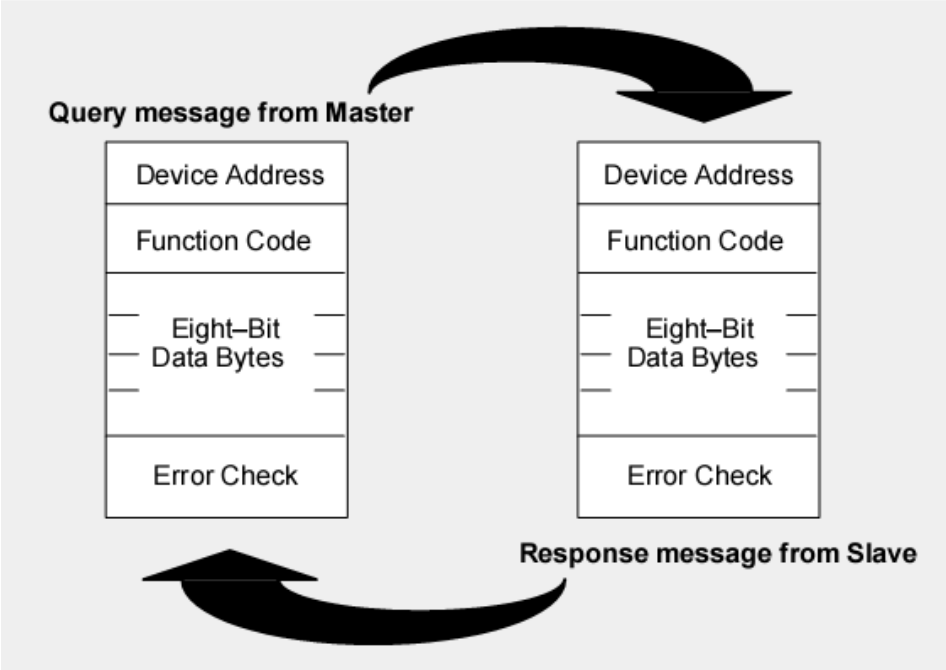
Function	Description
1	Read single/multiple coil (discrete output) status. Function 1 returns the on/off status of any number of output points (Qs).
2	Read single/multiple contact (discrete input) status. Function 2 returns the on/off status of any number of input points (Is).
3	Read single/multiple holding registers. Function 3 returns the contents of V memory. Holding registers are word values under Modbus and allow you to read up to 120 words in one request.
4	Read single/multiple input registers. Function 4 returns Analog Input values.
5	Write single coil (discrete output). Function 5 sets a discrete output point to the specified value. The point is not forced and the program can overwrite the value written by the Modbus request.
6	Write single holding register. Function 6 writes a single holding register value to the V memory of the S7-200.
15	Write multiple coils (discrete outputs). Function 15 writes the multiple discrete output values to the Q image register of the S7-200. The starting output point must begin on a byte boundary (for example, Q0.0 or Q2.0) and the number of outputs written must be a multiple of eight. This is a restriction for the Modbus Slave Protocol instructions. The points are not forced and the program can overwrite the values written by the Modbus request.
16	Write multiple holding registers. Function 16 writes multiple holding registers to the V memory of the S7-200. There can be up to 120 words written in one request.

Sau mã hàm là địa chỉ vùng nhớ và chiều dài vùng nhớ, dử liệu ghi vào hay đọc về. Địa chỉ vùng nhớ S7-200 cho trong bảng sau

000001 to 000128 ứng với ngõ ra số Q0.0 đến Q15.7
010001 to 010128 ứng với ngõ vào số I0.0 đến I15.7

030001 to 030032 ứng với ngõ vào analog AIW0 đến AIW62
040001 to 04xxxx ứng với vùng nhớ VW

Khi truyền đi ta bỏ byte đầu tiên của địa chỉ, chỉ truyền hai byte cuối bớt 1, như vậy muốn đọc 4 ngõ ra Q1.0 đến Q1.7 , vùng FUNCTION DATA sẽ là 0100080004



Ví dụ: đọc 37 bit từ địa chỉ 20 của slave địa chỉ 17

Bản tin từ master

QUERY

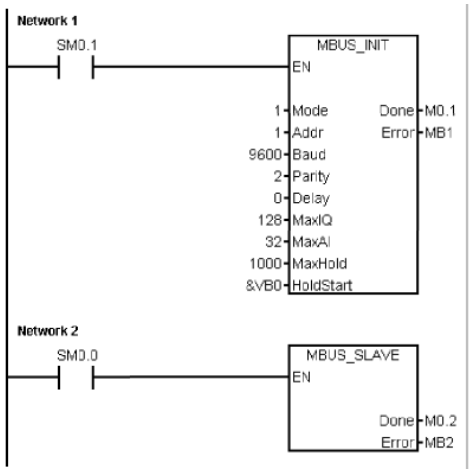
Field Name	Example (Hex)
Slave Address	11
Function	01
Starting Address Hi	00
Starting Address Lo	13
No. of Points Hi	00
No. of Points Lo	25
Error Check (LRC or CRC)	—

Bản tin từ slave

RESPONSE

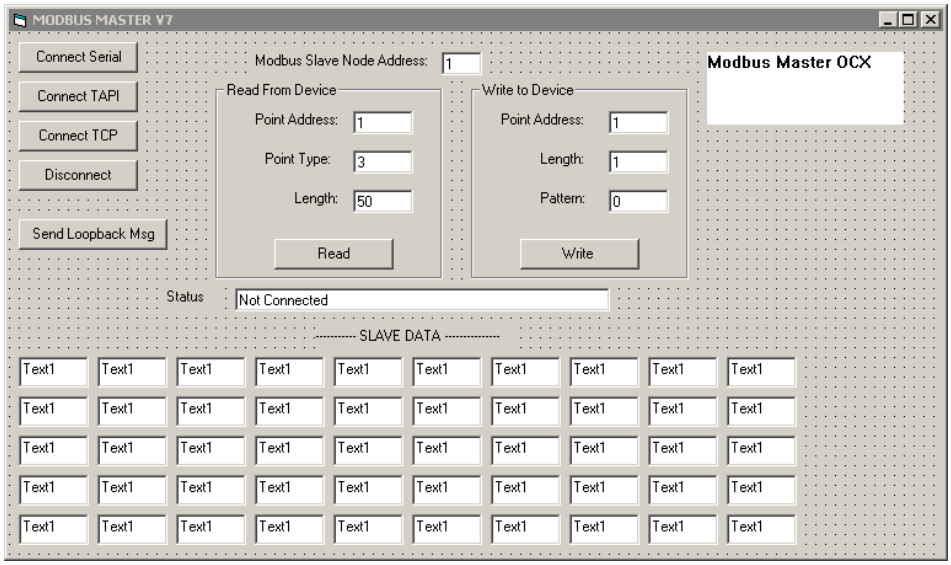
Field Name	Example (Hex)
Slave Address	11
Function	01
Byte Count	05
Data (Coils 27–20)	CD
Data (Coils 35–28)	6B
Data (Coils 43–36)	B2
Data (Coils 51–44)	0E
Data (Coils 56–52)	1B
Error Check (LRC or CRC)	—

S7-200 slave có thể giao tiếp với thiết bị khác theo giao thức chuẩn Modbus của Modicon nhờ vào thư viện Modbus Protocol Library, có hai lệnh MBUS_INIT và MBUS_SLAVE.




```
Network 1
          //Initialize the Modbus Slave Protocol on the
          //first scan. Set the slave address to 1, set
          // port 0 to 9600 baud with even parity, all
          //access to all I, Q and AI values, allow
          //access to 1000 holding registers (2000
          // bytes) starting at VB0.
LD        SM0.1
CALL      MBUS_INIT,1,1,9600,2,0,128,32,1000,
          &VB0,M0.1,MB1
Network 2
          //Execute the Modbus Slave Protocol on
          //every scan.
LD        SM0.0
CALL      MBUS_SLAVE,M0.2,MB2
```

Máy tính giao tiếp chuẩn Modbus thông qua thư viện MbMasterV7 Activex Control module mbmasterv7.OCX giúp tạo bản tin gởi đến slave và xử lý bản tin từ slave. Giao tiếp có thể thực hiện qua cổng COM hay cổng LAN. Sau đây là chương trình mẫu trong VB




```
'Global Data Definitions
Dim MyHandle As Long          'Handle to Connection
Dim MyStatus As Integer       'Status returned from mbMasterV7 Control
Dim Slave As Integer          'Slave, Cmd, Address, & Length
Dim Cmd As Integer            'represent last message sent to modbus
Dim Address As Long
Dim Length As Integer
Dim LoopbackMsg(20) As Byte

' Generic function to display status value returned  from PollModbus, WriteModbus
Public Sub show_status(ErrCode As Integer)
   If (ErrCode = 0) Then
      STATUS.Text = "OK"
   ElseIf (ErrCode < 255) Then
      STATUS.Text = "Slave Device Exception Response"
   ElseIf (ErrCode = 256) Then
      STATUS.Text = "Invalid Connection Handle"
   ElseIf (ErrCode = 257) Then
      STATUS.Text = "Message Overrun"
   ElseIf (ErrCode = 258) Then
      STATUS.Text = "Invalid Point Address"
   ElseIf (ErrCode = 259) Then
      STATUS.Text = "Invalid Slave Node Address"
   ElseIf (ErrCode = 260) Then
      STATUS.Text = "Invalid Length"
   ElseIf (ErrCode = 261) Then
      STATUS.Text = "Unsupported Modbus Command"
   ElseIf (ErrCode = 263) Then
      STATUS.Text = "Slave Device Time-Out"
```

```
        ElseIf (ErrCode = 264) Then
           STATUS.Text = "Invalid Transmission Mode"
        ElseIf (ErrCode = 265) Then
           STATUS.Text = "Invalid CRC In Slave Response"
        ElseIf (ErrCode = 266) Then
           STATUS.Text = "Connection Not Established"
        ElseIf (ErrCode = 267) Then
           STATUS.Text = "Invalid Slave Response"
        ElseIf (ErrCode = 271) Then
           STATUS.Text = "Demo Time Expired"
        ElseIf (ErrCode = 272) Then
           STATUS.Text = "Invalid modbus/TCP Command"
        End If

    End Sub

    ' Hide the contrtol when the form loads
    Private Sub Form_Activate()
       MbMasterV71.HideControl
    End Sub
    ' Handler for the CONNECT SERIAL Button
    Private Sub ConnectSerial_Click()
       ' Connect to COMM Port
       MbMasterV71.BaudRate = 9600  '9600 Baud
       MbMasterV71.Parity  =  0          '0=NOPARITY,  1=ODDPARITY,  2=EVENPARITY,
3=MARKPARITY, 4=SPACEPARITY
       MbMasterV71.DataBits = 8     '8 DataBits
       MbMasterV71.StopBits  =  0  '0  =  ONESTOPBIT,  1  =  ONE5STOPBITS,  2  =
TWOSTOPBITS
       MbMasterV71.TimeOut = 2000   '2000 msec
       MbMasterV71.TransmissionMode = 1 '0=ASCII, 1=RTU

       MyHandle = MbMasterV71.ConnectSerial(1)  ' Connect to COMM Port 1
       If MyHandle > 0 Then
          ' Connection was successful
          ' (This example only allows a single connection)
          ' Disable All Connection Buttons
          ' Enable the Read, Write & Disconnect Buttons
          ConnectSerial.Enabled = False
          ConnectTAPI.Enabled = False
          ConnectTCP.Enabled = False
          Disconnect.Enabled = True
          LoopBackTst.Enabled = True
          STATUS.Text = "Connected"
          READMODBUS.Enabled = True
          WRITEMODBUS.Enabled = True
       Else
          'Connection Attempt Failed
```

```
        '(Another application must have control of the COM Port)
        STATUS.Text = "Not Connected"
    End If
End Sub

' Handler for the CONNECT TAPI Button
Private Sub ConnectTAPI_Click()
Dim nTAPIDevices As Long
Dim TAPIDevice As String

    'Go through the motions of getting the TAPI Device List
    nTAPIDevices = MbMasterV71.NumberOfTAPIDevices()
    TAPIDevice = MbMasterV71.GetTAPIDeviceName(0)
    'Setup the phone number to dial
    MbMasterV71.PhoneNumber = "645-5966"
    'Dial the call
    MyHandle = MbMasterV71.DialTAPIDevice(0)
    If MyHandle > 0 Then
        'Call should be in progress now Don't enable the Read & Write Buttons
        'until we get the CallEstablished Event
        STATUS.Text = "Connecting"
        ConnectSerial.Enabled = False
        ConnectTAPI.Enabled = False
        ConnectTCP.Enabled = False
        Disconnect.Enabled = False
        LoopBackTst.Enabled = False
        READMODBUS.Enabled = False
        WRITEMODBUS.Enabled = False
    Else
        STATUS.Text = "Not Connected"
    End If

End Sub

' Handler for the CONNECT TCP Button
Private Sub ConnectTCP_Click()
    ' Select the Device to connec to
    ' In this case use the IP Loopback address to
    ' connect to the local machine
    MbMasterV71.TCPDevice = "127.0.0.1"
    MyHandle = MbMasterV71.ConnectModbusTCP(502)
    If MyHandle > 0 Then
        'Connection should be in progress now
        'Don't enable the Read & Write Buttons
        'until we get the CallEstablished Event
        STATUS.Text = "Connecting"
        ConnectSerial.Enabled = False
        ConnectTAPI.Enabled = False
```

```
        ConnectTCP.Enabled = False
        Disconnect.Enabled = False
        LoopBackTst.Enabled = False
        READMODBUS.Enabled = False
        WRITEMODBUS.Enabled = False
    Else
        STATUS.Text = "Not Connected"
    End If

End Sub


' ConnectionEstablished Event Handler
' Initiated from either ConnectModbusTCP() or DialTAPIDevice()
Private Sub MbMasterV71_ConnectionEstablished(ByVal hConnect As Long)

    'Enable the Disconnect Button, Enable Read & Write Buttons
    ConnectSerial.Enabled = False
    ConnectTAPI.Enabled = False
    ConnectTCP.Enabled = False
    Disconnect.Enabled = True
    LoopBackTst.Enabled = True
    STATUS.Text = "Connected"
    READMODBUS.Enabled = True
    WRITEMODBUS.Enabled = True

End Sub

' ConnectionDropped Event Handler
Private Sub MbMasterV71_ConnectionDropped(ByVal hConnect As Long)
    'Either the TCP or TAPI connection attempt failed
    'or something has happened to abort the connection after it
    'has been established for a while.
    'In either case we're now disconnected so enable the
    'buttons accordingly
    MyHandle = -1
    ConnectSerial.Enabled = True
    ConnectTAPI.Enabled = True
    ConnectTCP.Enabled = True
    Disconnect.Enabled = False
    LoopBackTst.Enabled = False
    STATUS.Text = "Not Connected"
    READMODBUS.Enabled = False
    WRITEMODBUS.Enabled = False

End Sub
```

```
' Handler for the DISCONNECT Button
Private Sub Disconnect_Click()
    'Tell the control to Disconnect
   MyStatus = MbMasterV71.Disconnect(MyHandle)
   MyHandle = -1
   'ReEnable the Connect Buttons for new connection attempt
   ConnectSerial.Enabled = True
   ConnectTAPI.Enabled = True
   ConnectTCP.Enabled = True
   Disconnect.Enabled = False
   LoopBackTst.Enabled = False
   STATUS.Text = "Not Connected"
   READMODBUS.Enabled = False
   WRITEMODBUS.Enabled = False
End Sub


' Handler for the READ Button
Private Sub READMODBUS_Click()
    'Get the Slave Node Address, Modbus Command, Address & Length
    ' from the appropriate Edit controls
   Slave = NODEADDRESS.Text
   Cmd = POINTTYPE.Text
   Address = READADDRESS.Text
   Length = READLENGTH.Text
    'We must remember these parameters so we can use
    'them in the ReadResponse method to make sure we
    'get what we ask for
    '
    'Initiate the Read Request
   MyStatus = MbMasterV71.PollModbus(MyHandle, Slave, Cmd, Address, Length)
   'Check the status to make sure the request went out.
   If MyStatus = 0 Then
      STATUS.Text = "Busy"
   Else
      show_status (MyStatus)
End If

End Sub

'Process the Slave Read Response Message
'Modbus_Master SlaveReadResponse Event Handler
Private Sub MbMasterV71_SlaveReadResponse(ByVal hConnect As Long)
Dim MyData As Long
Dim i As Integer
    ' Read the data returned from the slave
    ' and update the text controls
```

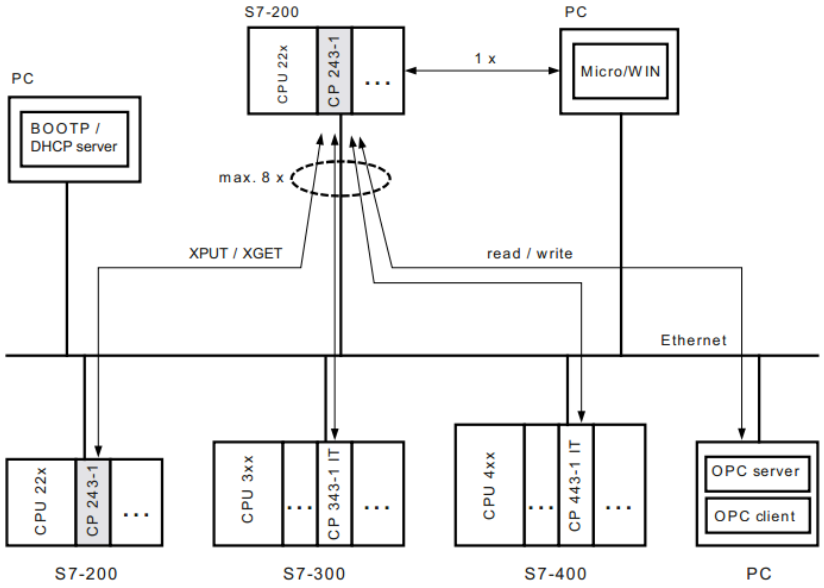
```
        For i = 0 To Length - 1
           MyStatus = MbMasterV71.ReadResults(hConnect, Slave, Cmd, Address + i,
MyData)
           show_status (MyStatus)
           If MyStatus = 0 Then
              Text1(i).Text = MyData
           Else
              Text1(i).Text = "Error"
           End If
        Next i
        ' For display purposes, flag the unread data as ?
        For i = Length To 49
           Text1(i).Text = "?"
        Next i

     End Sub


     ' Handler for the WRITE Button
     Private Sub WRITEMODBUS_Click()
     Dim IsRegister As Boolean
     Dim i As Integer
     Dim junk As Integer
        'Get the Slave Node Address, Modbus Command, Address & Length
        ' from the appropriate Edit controls
        Slave = NODEADDRESS.Text
        Address = WRITEADDRESS.Text
        Length = WRITELENGTH.Text
        'Make sure we do not overrun the message
        ' buffer allocated within the control
        If Length > 200 Then
           Length = 200
        End If
        ' Fill the write buffer with the specified number
        ' of values
        For i = 0 To Length - 1
           junk = MbMasterV71.FillWriteBuffer(MyHandle, i, PATTERN.Text)
        Next i

        ' figure out which modbus command to use
        If POINTTYPE.Text < 2 Then
           If Length = 1 Then
              Cmd = 5 'write single coil
           Else
              Cmd = 15 'write multiple coils
           End If
        Else
           If Length = 1 Then
```

```
            Cmd = 6 'write single register
          Else
            Cmd = 16 'write multiple registers
          End If
        End If

        ' Initiate the Write Request Message
        MyStatus = MbMasterV71.WRITEMODBUS(MyHandle, Slave, Cmd, Address,
Length)
        ' Make sure the write request was transmitted
        If MyStatus = 0 Then
          STATUS.Text = "Busy"
        Else
          show_status (MyStatus)
      End If
      End Sub

      'Process the Slave Write Response Message
      'Modbus_Master SlaveWriteResponse Event Handler
      Private Sub MbMasterV71_SlaveWriteResponse(ByVal hConnect As Long)
        'read the results of the write request
        MyStatus = MbMasterV71.WriteResults(hConnect, Slave, Cmd, Address, Length)
        ' and update the status display
        show_status (MyStatus)
      End Sub

      ' Handler for the Loopback Test Button
      Private Sub LoopBackTst_Click()
      ' Use the User Message API to generate a modbus loopback message and send to the
slave device
        'Get the Slave Node Address from the appropriate Edit controls
        Slave = NODEADDRESS.Text
        Cmd = 8
        MyStatus = MbMasterV71.FillUserMsgBuffer(MyHandle, 0, Slave)
        MyStatus = MbMasterV71.FillUserMsgBuffer(MyHandle, 1, Cmd)
        MyStatus = MbMasterV71.FillUserMsgBuffer(MyHandle, 2, 0)
        MyStatus = MbMasterV71.FillUserMsgBuffer(MyHandle, 3, 0)
        MyStatus = MbMasterV71.FillUserMsgBuffer(MyHandle, 4, 0)
        MyStatus = MbMasterV71.FillUserMsgBuffer(MyHandle, 5, 0)
          MyStatus = MbMasterV71.SendUserMsg(MyHandle, 6)
        'Check the status to make sure the request went out.
        If MyStatus = 0 Then
          STATUS.Text = "Busy"
        Else
          show_status (MyStatus)
        End If
      End Sub
```

```
'Process the Slave Loopback Response Message
'Modbus_Master UserMsgResponse Event Handler
Private Sub MbMasterV71_UserMsgResponse(ByVal hConnect As Long, ByVal
NumberOfBytes As Long)
Dim temp As Integer
  If NumberOfBytes > 0 Then
    For i = 0 To NumberOfBytes - 1
      MyStatus = MbMasterV71.ReadUserMsgResponse(hConnect, i, temp)
      LoopbackMsg(i) = temp
    Next i
    STATUS.Text = LoopbackMsg
    End If

End Sub
```

9.31 GIAO TIẾP QUA MẠNG ETHERNET

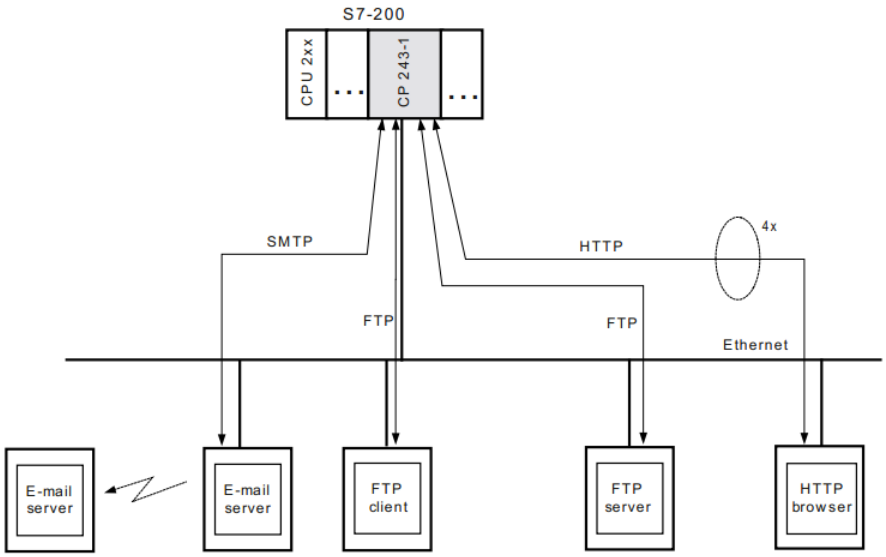
S7-200 có thể giao tiếp qua mạng Ethernet khi dùng module truyền thông CP 243-1 IT



Cấu hình mạng Ethernet

S7-200 truyền thông với các PLC khác và với OPC Server cài trên máy tính dùng các lệnh truyền thông S7 (XPUT, XGET, READ, WRITE) tối đa 8 kênh. Ngoài ra nó có thể gởi email (qua giao thức SMTP đến email server, là FTP server hay client trao đổi file với thiết bị mạng dùng giao thức FTP và là server cho giao thức HTTP kết nối với browser. Cấu hình CP243-1 được thực hiện nhờ Internet Wizard của Microwin, ta phải ấn định trước các thông

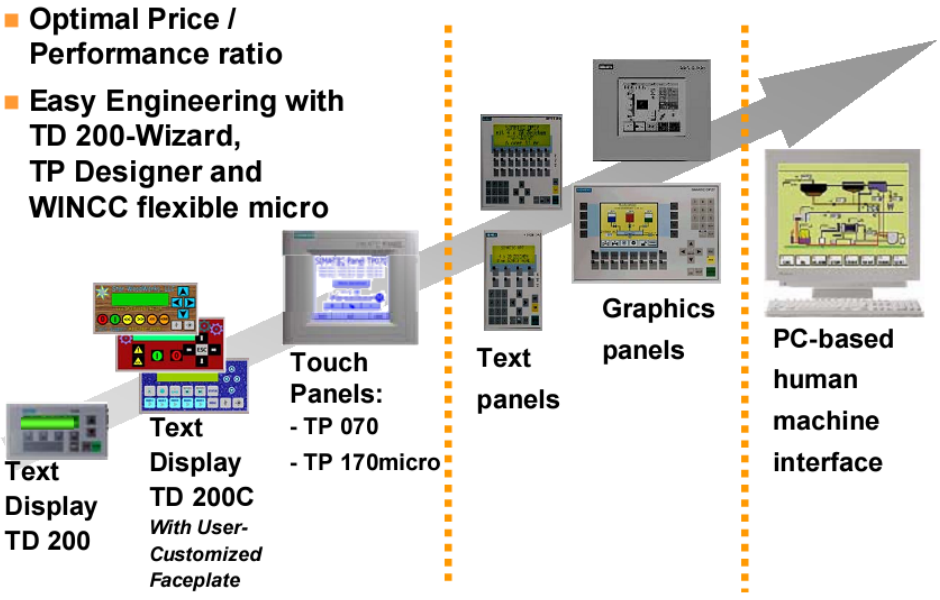
số mạng như địa chỉ IP tĩnh (ví dụ 172.28.24.2), Subnet Mask



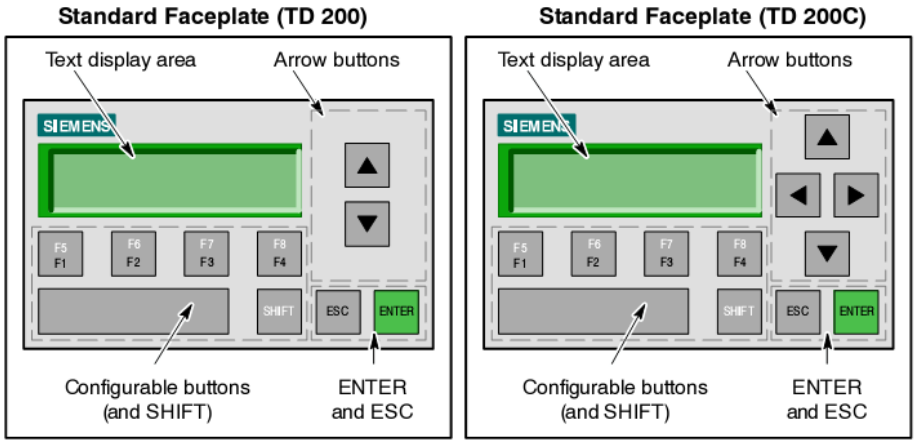
CẤU HÌNH CÔNG NGHỆ THÔNG TIN IT

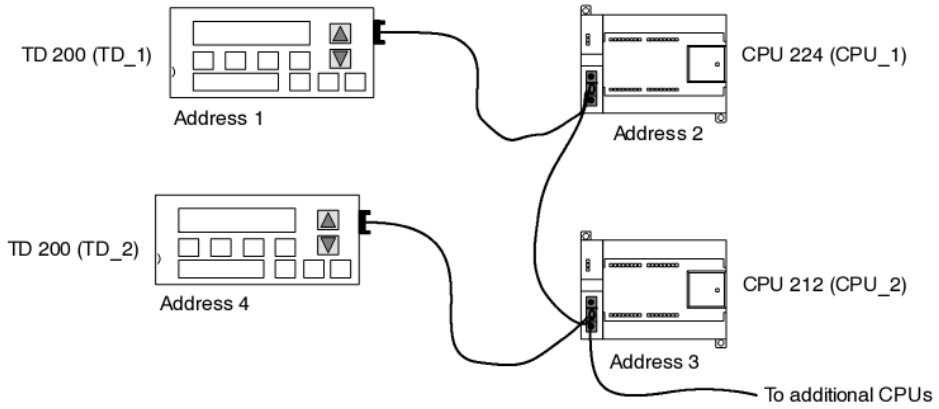
(255.255.255.0) , gateway (0.0.0.0)…cài đặt cấu hình email, FTP và HTTP

9.32 LẬP TRÌNH MÀN HÌNH TD200



TD200 là màn hình hiển thị ký tự và có các phím để giao tiếp với S7-200. Màn hình tinh thể lỏng có 2 hàng ký tự với tổng cộng 40 ký tự để hiển thị và thay đổi nội dung các ô nhớ V của S7-200, các cảnh báo, thời gian. Có thể dùng nhiều tập font ký tự như Latin, Cyrillic, Do thái, Ả rập, Thổ, Hy lạp, Baltic. Menu và Prompt thể hiện dưới các ngôn ngữ Anh, Pháp, Ý, Đức, Tây ban nha, Trung quốc. Thông qua các phím trên TD200, ta có thể bật các bit của S7-200 ON/OFF. Lập trình TD200 dùng TD200 Wizard của Microwin. Có thể ghép một TD200 với một hay nhiều S7-200. Một S7-200 có thể ghép với nhiều TD200, mỗi TD200 có địa chỉ cụ thể và liên kết với S7-200 có địa chỉ cụ thể thông qua khối dữ liệu vùng nhớ tham số V. Thông thường máy tính lập trình có địa chỉ là 0, TD200 địa chỉ 1 và PLC địa chỉ 2





TD200 Wizard giúp cài đặt khối dữ liệu để S7-200 giao tiếp TD200:

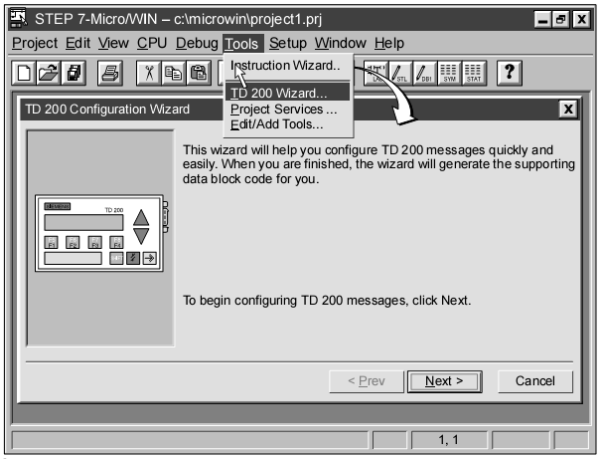
- Chọn loại TD200 (version 2.1, 3.0, C), tùy theo loại TD200, wizard sẽ diễn tiến khác nhau
- Ngôn ngữ và font chữ
- Cài đặt password (0000 đến 9999) và các chức năng như thay đổi ngày giờ PLC, cưỡng bức ngõ vào ra PLC, thay đổi nội dung ô nhớ, ghi thẻ nhớ
- Chọn bit M bị tác động khi nhấn các phím F1..F4 và ShiftF1.. ShiftF4. Các phím có thể là loại set hay on/off.
- Chọn màn hình 20 ký tự (hai cảnh báo) hay 40 ký tự (một cảnh báo)
- Cảnh báo gồm bản tin và dữ liệu từ vùng nhớ V của PLC (có thể có đến 6 ô nhớ cho một cảnh báo), tối đa 80 cảnh báo. Cảnh báo xuất hiện khi một bit cho phép tương ứng ON (message enabled flag). Có thể cài đặt bit xác nhận cảnh báo (acknowledgement bit) lúc đó người dùng phải bấm một phím nào đó để bit này on . Dữ liệu kèm theo có thể được thay đổi bởi người vận hành (editable) bằng cách nhấn enter, thay đổi dữ liệu bằng phím mũi tên, kết thúc bấm enter , bit edit notification sẽ thay đổi báo đã edit xong.
- Cài đặt menu người dùng: có thể có 8 lựa chọn, mỗi lựa chọn có 8 màn hình, mỗi màn hình bao gồm message và nội dung ô nhớ, chọn menu dùng các phím mũi tên, enter, esc

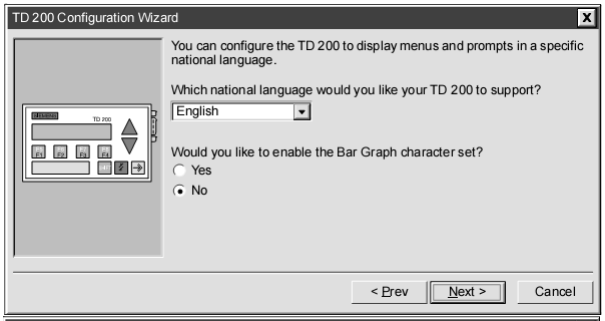
- Chọn vùng nhớ V chứa dữ liệu

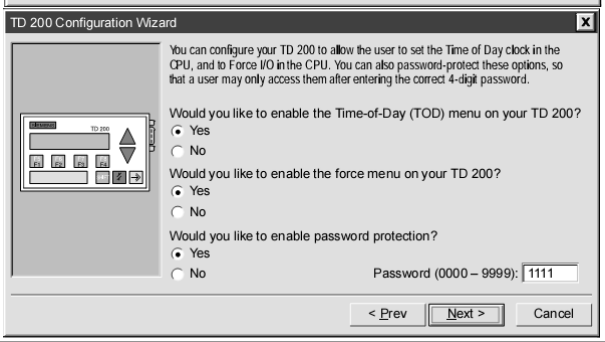
- Sau khi kết thúc, wizard tạo các chương trình con, khối dữ liệu tham số, bảng ký hiệu.

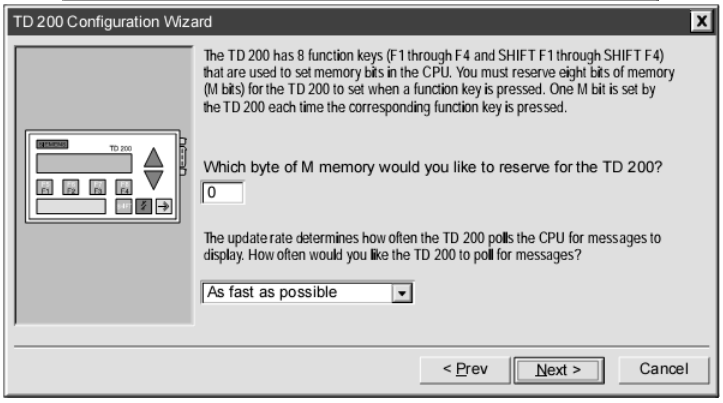
- Sau khi lập trình PLC và download, ta kết nối TD200 với PLC, vào màn hình Diagnostic Menu – TD Setup để chọn địa chỉ PLC, TD200, địa chỉ vùng nhớ tham số TD200, vận tốc truyền.

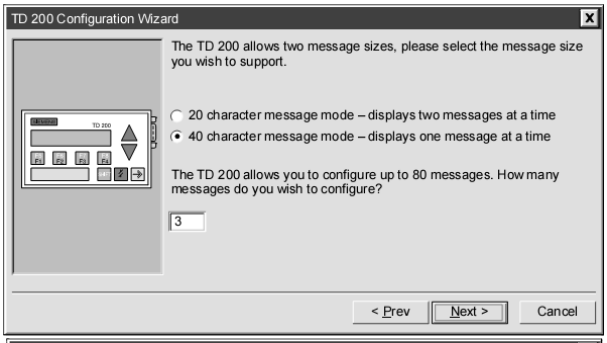
Ví dụ: lập trình cho TD200 V2.1 dùng các phím F để set M0.x, hiển thị ba message

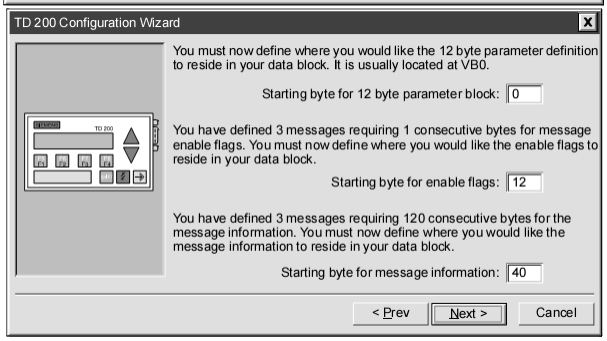

STEP 7-Micro/WIN – c:\microwin\project1.prj
Project Edit View CPU Debug Tools Setup Window Help
Instruction Wizard..
TD 200 Wizard...
Project Services ...
Edit/Add Tools...
TD 200 Configuration Wizard
This wizard will help you configure TD 200 messages quickly and easily. When you are finished, the wizard will generate the supporting data block code for you.
To begin configuring TD 200 messages, click Next.
< Prev
Next >
Cancel
1, 1


TD 200 Configuration Wizard
You can configure the TD 200 to display menus and prompts in a specific national language.
Which national language would you like your TD 200 to support?
English
Would you like to enable the Bar Graph character set?
Yes
No
< Prev
Next >
Cancel

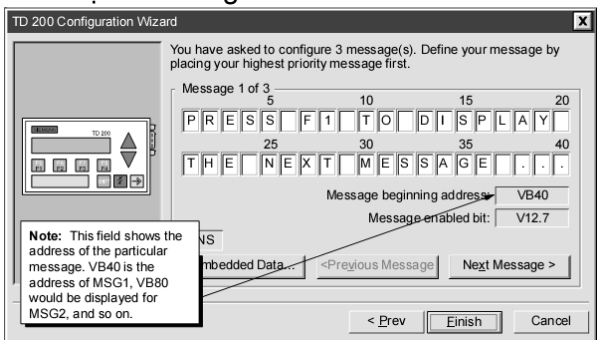

TD 200 Configuration Wizard
You can configure your TD 200 to allow the user to set the Time of Day clock in the CPU, and to Force I/O in the CPU. You can also password-protect these options, so that a user may only access them after entering the correct 4-digit password.
Would you like to enable the Time-of-Day (TOD) menu on your TD 200?
Yes
No
Would you like to enable the force menu on your TD 200?
Yes
No
Would you like to enable password protection?
Yes
No
Password (0000 – 9999): 1111
< Prev
Next >
Cancel


TD 200 Configuration Wizard
The TD 200 has 8 function keys (F1 through F4 and SHIFT F1 through SHIFT F4) that are used to set memory bits in the CPU. You must reserve eight bits of memory (M bits) for the TD 200 to set when a function key is pressed. One M bit is set by the TD 200 each time the corresponding function key is pressed.
Which byte of M memory would you like to reserve for the TD 200?
0
The update rate determines how often the TD 200 polls the CPU for messages to display. How often would you like the TD 200 to poll for messages?
As fast as possible
< Prev
Next >
Cancel

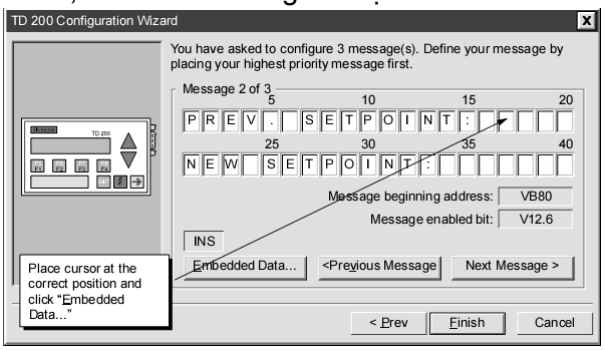


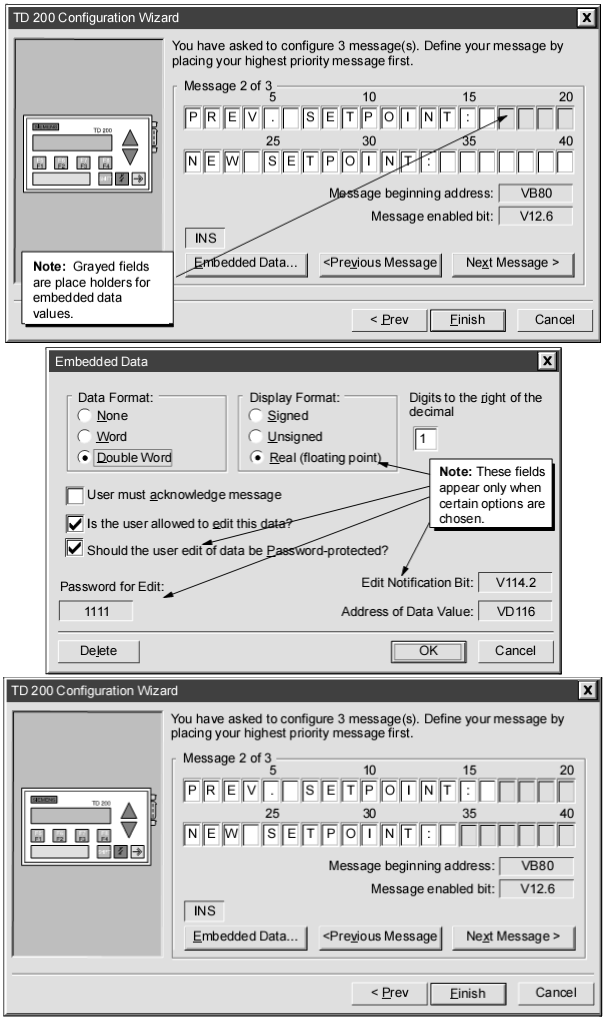


Khi V12.7 on thì hiện message 1

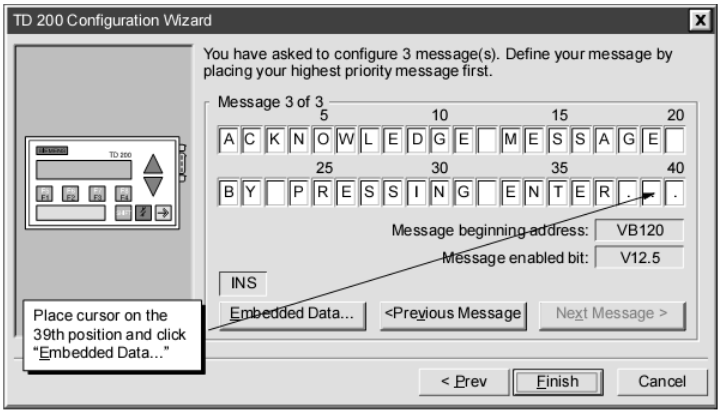


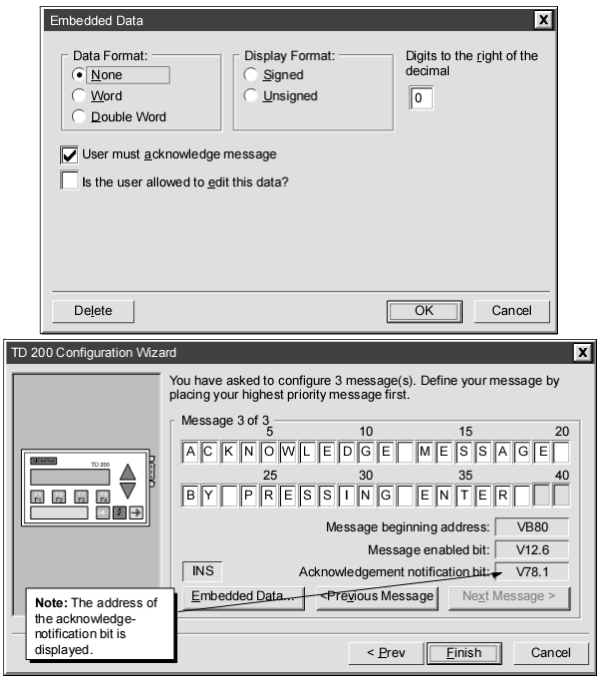
Message 2 hiển thị khi V12.6 on, hiển thị ô nhớ VW98 và cho thay đổi ô nhớ VD116, bit V114.2 on ghi nhận edit





Message 3 hiển thị khi V12.5 on, yêu cầu người vận hành ghi nhận bằng phím Enter, lúc đó bit V78.1 on





Wizard tạo ra khối dữ liệu TD200_Block 0 từ địa chỉ VB0 trong PLC để giao tiếp với TD200

```
// BEGIN TD200_BLOCK 0
// (Comments within this block should not be edited or removed)
VB0 'TD' // TD 200 Identification
VB2 16#10 // Set Language to English, set Update to As fast as possible
VB3 16#71 // Set the display to 40 character mode; Up key V3.2; Down key V3.3
VB4 3 // Set the number of messages
VB5 0 // Set the Function Keys notification bits to M0.0 – M0.7
VW6 40 // Set the starting address for messages to VW40
VW8 12 // Set the starting address for message enable bits to VW12
VW10 1111 // Global Password
// MESSAGE 1
// Message Enable Bit V12.7
VB40 'PRESS F1 TO DISPLAY THE NEXT MESSAGE ...'
// MESSAGE 2
// Message Enable Bit V12.6
VB80 'PREV. SETPOINT: '
VB96 16#00 // No Edit; No Acknowledgement; No Password
VB97 16#11 // Signed Word; 1 Digits to the right of the decimal;
VW98 16#00 // Embedded Data Value: Move data for display here.
VB100 'NEW SETPOINT: '
VB114 16#18 // Edit Notification V114.2; No Acknowledgement; Edit Requires Passw
VB115 16#51 // Real Double Word; 1 Digits to the right of the decimal;
VD116 16#0000 // Embedded Data Value: Move data for display here.
// MESSAGE 3
// Message Enable Bit V12.5
VB120 'ACKNOWLEDGE MESSAGE BY PRESSING ENTER:'
```

```
VB158 16#01 // No Edit; Acknowledgement Notification V158.1; No Password
VB159 16#00 // No Data; 0 Digits to the right of the decimal;
// END TD200_BLOCK 0
```

Bây giờ lập trình PLC

```
NETWORK 1
LD SM0.1 // if this is the first scan
MOVB 16#80, VB12 // ...enable the first message
MOVB 0, MB0 // ...clear all function key bits
NETWORK 2
LD M0.0 // if F1 has been pressed
MOVB 16#40, VB12 // ...enable message 2 for display
R M0.0, 1 // ...reset F1 key M bit
NETWORK 3
LD V114.2 // if new setpoint edit bit is set
R V114.2, 1 // ...reset edit bit
MOVR VD116, AC0 // ...get edited real value
*R 10.00000, AC0 // ...times 10 for scaling
TRUNC AC0, AC1 // ... convert to an integer
MOVW AC1, VW98 // ... update prev. setpoint value
MOVB 16#20, VB12 // ...enable message 3 for display
```

```
NETWORK 4
LD V158.1 // if message 3 acknowledge bit is set
R V158.1, 1 // ...reset message 3 ack bit
MOVB 16#80, VB12 // ...enable message 1 for display
NETWORK 5
LD M0.1 // if F2 has been pressed
MOVB 16#E0, VB12 // ...enable all 3 messages at once
R M0.1, 1 // ...reset F2 key M bit
NETWORK 6
LD M0.2 // if F3 has been pressed
MOVB 0, VB12 // ...disable all messages
R M0.2, 1 // ...reset F3 key M bit
NETWORK 7
MEND
```

Download chương trình và khối dữ liệu xuống PLC, kết nối PLC với TD200, cài đặt thông số kết nối, TD200 sẽ đi tìm khối dữ liệu và truyền thông với S7-200, dùng các phím sau:

- bấm F1 đi đến message setpoint
- bấm Enter edit setpoint và ghi nhận message
- bấm F2 cho phép cả ba message
- bấm F3 cấm cả ba message

Ngoài TD200, S7-200 còn có thể giao tiếp với các thiết bị HMI khác như TP070/170/177, OP3/73.

-------O--------

Bài tập gợi ý

1/Nghiên cứu các thiết bị HMI của OMRON và SIEMENS
2/ Nghiên cứu giao thức truyền Modbus
3/ Lập trình giao tiếp với máy tính