1. GIỚI THIỆU PLC S7-200

The S7-200 series is a line of micro-programmable logic controllers (Micro PLCs) that can control a variety of automation applications. Figure 1-1 shows an S7-200 Micro PLC. The compact design, expandability, low cost, and powerful instruction set of the S7-200 Micro PLC make a perfect solution for controlling small applications. In addition, the wide variety of CPU sizes and voltages provides you with the flexibility you need to solve your automation problems.

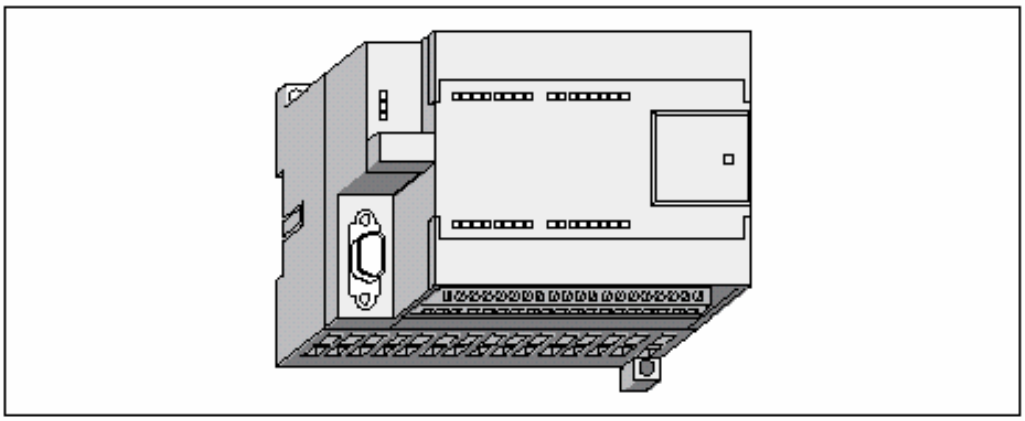


Figure 1-1 S7-200 Micro PLC

PLC SIEMENS thế hệ S7-200 là PLC loại nhỏ (Micro PLC), có thể điều khiển hàng loạt các ứng dụng khác nhau trong tự động hóa. (Hình 1-1). Với cấu trúc nhỏ gọn, có khả năng mở rộng, giá rẻ và một tập lệnh mạnh, PLC S7-200 là một lời giải hoàn hảo cho các bài toán tự động loại nhỏ. Thêm vào đó là sự phong phú về chủng loại, kích cỡ cũng như các thông số về điện (điện áp AC, DC, dòng, . . .) càng cho phép người sử dụng cơ động hơn trong việc giải quyết các vấn đề tự động của mình.

Nói về các chủng loại phong phú của PLC S7-200, chúng ta dựa trên nhiều tiêu chí khác nhau:

- Nguồn nuôi: điện áp một chiều 24V, điện áp xoay chiều 220V, 110V.
- Đầu vào 24VDC: sink & source.
- Đầu ra 24VDC hoặc Rơ le.
- Các bộ xử lý trung tâm (CPU) khác nhau: S7-210, 212, 214, 215, 216, 221, 222, 224.

Sau đây là bảng so sánh các CPU thông dụng nhất:

Đặc trưng kỹ thuật	**CPU 221**	**CPU 212**	**CPU 222**	
Bộ nhớ chương trình	4 kB	1 kB	4 kB	
Bộ nhớ dữ liệu	2 kB	1 kB	2 kB	
Backup số liệu	50 giờ	50 giờ	50 giờ	
Thời gian thực hiện 1024 lệnh nhị phân	0.37 ms	1.2 ms	0.37 ms	
Xử lý số thực	có	không	có	
Điều khiển PID	có	không	có	
Số bộ định thời	256	64	256	
Số bộ đếm	256	64	256	
Bộ đếm tốc độ cao	4x30kHz	1x2kHz	4x30kHz	
Số ngắt thời gian	2 (1-255ms)	1 (5-255ms)	2 (1-255ms)	
Số ngắt phần cứng	4	1	4	
Số đầu vào / ra có sẵn	6/4	8/6	8/6	
Số module cực đại	0	2	2	
Số đầu vào / ra cực đại	6/4	40/38	24/22	
Số đầu vào / ra tương tự cực đại	0/0	6/4	12/10	
Đầu ra xung	2x20kHz	0	2x20kHz	
Cổng truyền thông	1xRS485	1xRS485	1xRS485	
Chiết áp tương tự	1	1	1	
Thời gian thực	tùy chọn	không	tùy chọn	
Kích thước (mm)	90x80x62	160x80x62	90x80x62	
Trọng lượng (kg)	0.27/0.31	0.39	0.27/0.39	

Đặc trưng kỹ thuật	**CPU 214**	**CPU 224**	**CPU 215**	**CPU 216**
Bộ nhớ chương trình	4 kB	8 kB	8 kB	8 kB
Bộ nhớ dữ liệu	4 kB	5 kB	5 kB	5 kB
Backup số liệu	190 giờ	190 giờ	190 giờ	190 giờ
Thời gian thực hiện 1024 lệnh nhị phân	0.8 ms	0.37 ms	0.8 ms	0.8 ms
Xử lý số thực	có	có	có	có
Điều khiển PID	có	có	có	có
Số bộ định thời	128	256	256	256
Số bộ đếm	128	256	256	256
Bộ đếm tốc độ cao	1x2kHz,	6x30kHz	1x2kHz,	1x2kHz,

	2x7kHz		2x20kHz	2x20kHz
Số ngắt thời gian	2 (5-255ms)	2 (1-255ms)	2 (5-255ms)	(5-255ms)
Số ngắt phần cứng	4	4	4	4
Số đầu vào / ra có sẵn	14/10	14/10	14/10	24/16
Số module cực đại	7	7	7	7
Số đầu vào / ra cực đại	62/58	62/58	62/58	64/64
Số đầu vào / ra tương tự cực đại	12/10	12/10	12/10	12/10
Đầu ra xung	2x4kHz	2x20kHz	2x4kHz	2x4kHz
Cổng truyền thông	1xRS485	1xRS485	2xRS485	2xRS485
Chiết áp tương tự	2	2	2	2
Thời gian thực	có	có	có	có
Kích thước (mm)	197x80x62	120x80x62	218x80x62	218x80x62
Trọng lượng (kg)	0.49	0.36/0.41	0.58	0.58

Các thành phần của một hệ thống với S7-200:

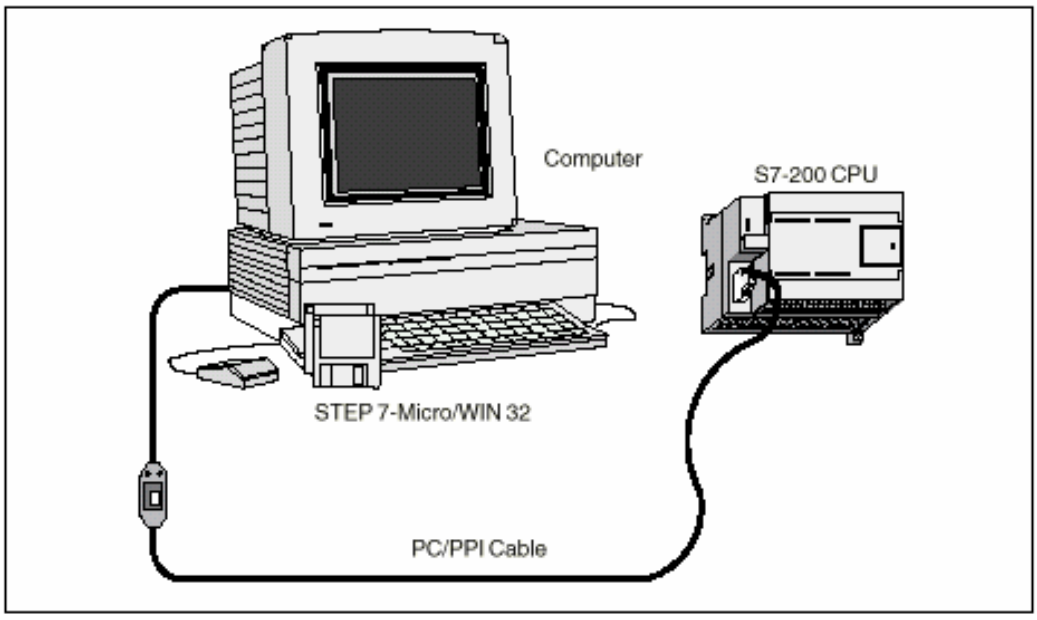


Figure 1-2 Components of an S7-200 Micro PLC System

Riêng S7-200 Micro PLC có thể bao gồm một mình S7-200 CPU hay có cả các module mở rộng.

S7-200 CPU

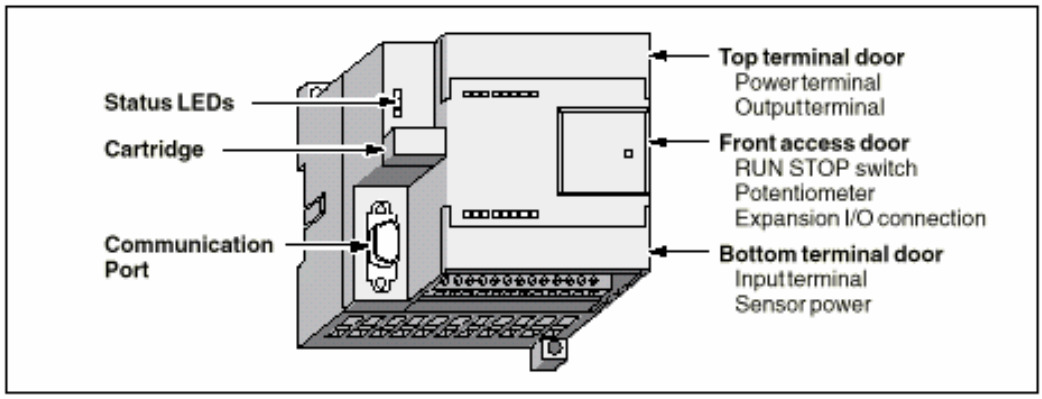


Figure 1-3 S7-200 CPU

S7-200 CPU tổng hợp cả bộ xử lý trung tâm, bộ nguồn nuôi và một số các đầu vào ra tạo nên một thiết bị có thể hoạt động độc lập.

- CPU thực thi chương trình và lưu giữ dữ liệu trong toàn bộ quá trình điều khiển.
- CPU có thể được tăng cường thêm các đầu vào ra cũng như các đầu vào ra tương tự bằng cách cắm thêm các module mở rộng.
- Bộ nguồn nuôi cung cấp cho bản thân CPU cũng như tất cả các module mở rộng.
- Các đầu vào đọc tín hiệu từ các thiết bị cảm biến (như các đầu dò và các công tắc); Các đầu ra điều khiển các thiết bị chấp hành như mô tơ, van, bơm, . . .
- Cổng truyền thông cho phép kết nối PLC với thiết bị lập trình hoặc các thiết bị khác (ví dụ các PLC khác).
- Các đèn hiệu trên CPU thông báo trạng thái CPU (ví dụ đang ở chế độ RUN hay STOP), trạng thái các cổng vào ra, lỗi hệ thống.
- Một số CPU có sẵn đồng hồ thời gian thực, một số khác có thể bổ sung tính năng này bằng cách cắm "CARTRIDGE thời gian thực".
- "CARTRIDGE EEPROM" cung cấp khả năng lưu giữ chương trình cũng như chuyển (copy) chương trình từ CPU này sang CPU khác.
- "CARTRIDGE pin" cho phép tăng cường thời gian lưu trữ dữ liệu trong bộ nhớ RAM.

Các Module mở rộng

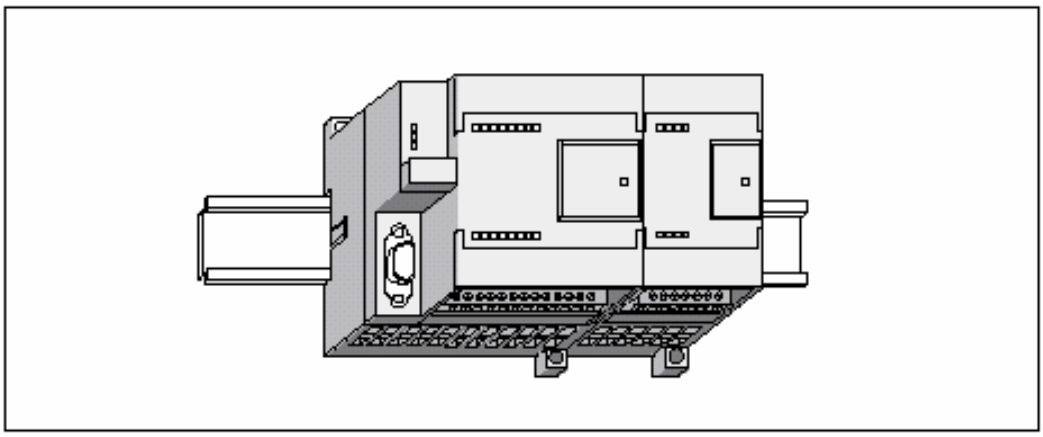
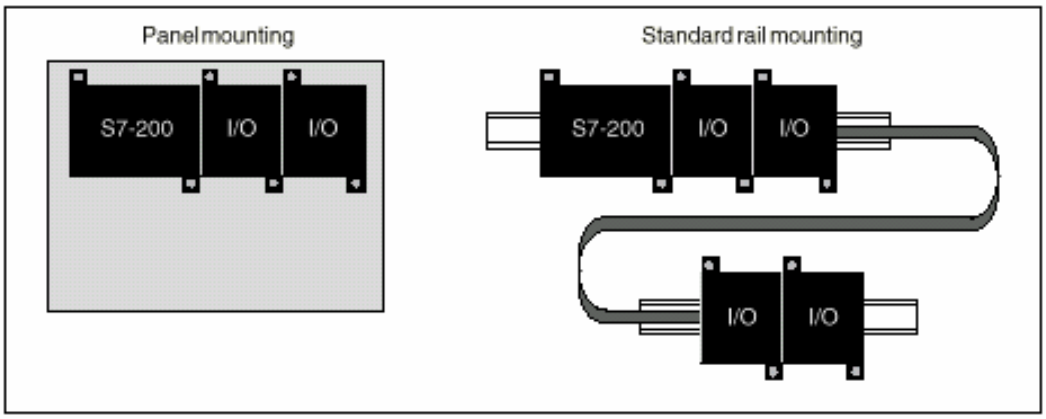


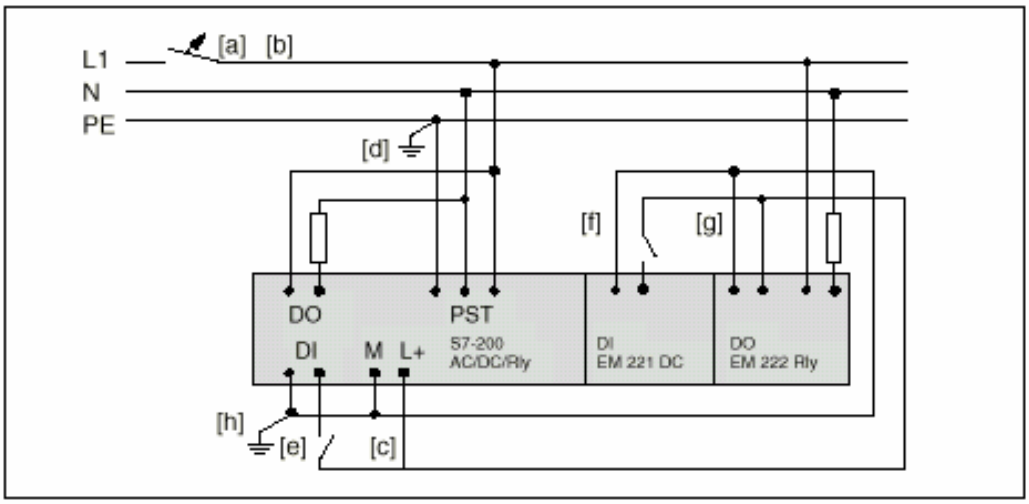
Figure 1-4 CPU with an Expansion Module

2. SỬ DỤNG PLC S7-200

Lắp đặt PLC trong hệ thống:



PLC sử dụng nguồn nuôi xoay chiều:



[a] Công tắc ngắt nguồn cho CPU, toàn bộ mạch vào và ra của PLC.

[b] Thiết bị chống quá dòng cho CPU, các mạch vào và ra. Có thể sử dụng cầu chì riêng cho từng phần (CPU, mạch vào, mạch ra) để bảo vệ tốt hơn.

[c] Bảo vệ quá dòng cho mạch vào không cần thiết nếu các đầu vào sử dụng nguồn 24VDC do PLC cung cấp. Nguồn này (gọi là nguồn cảm biến) đã được thiết kế chống ngắn mạch.

[d] Nối đầu đấu mát của PLC vào điểm nối đất gần nhất để chống nhiễu. Tất cả các đầu đấu mát trong một hệ thống nên được đấu vào cùng một điểm. Tốt nhất nên sử dụng dây 14 AWG hay dây 1.5 mm▒.

[e] Nguồn 24VDC do PLC cung cấp (nguồn cảm biến) có thể được sử dụng cho mạch các đầu vào.

[f] Nguồn 24VDC do PLC cung cấp (nguồn cảm biến) có thể được sử dụng cho mạch các đầu vào mở rộng.

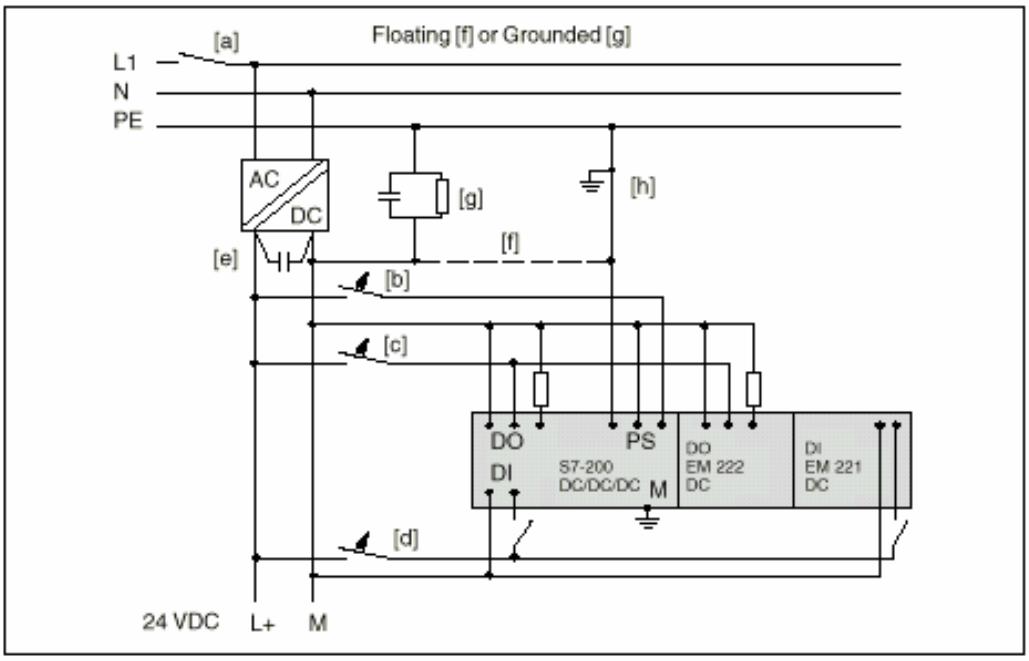
[g] Nguồn 24VDC do PLC cung cấp (nguồn cảm biến) có thể được sử dụng nuôi các module ra mở rộng.

(Nguồn cảm biến này đã được thiết kế chống ngắn mạch.)

[h] Trong đa số các trường hợp, nối đất đầu M của nguồn cảm biến 24VDC này là một trong những cách chống nhiễu tốt nhất.

PLC sử dụng nguồn nuôi một chiều:

[a] Công tắc ngắt nguồn cho CPU, toàn bộ mạch vào và ra của PLC.



[b] Thiết bị bảo vệ quá dòng cho CPU.

[c] Thiết bị bảo vệ quá dòng cho mạch vào.

[d] Thiết bị bảo vệ quá dòng cho mạch ra.

[e] Cần đảm bảo nguồn một chiều có đủ độ "cứng" cần thiết nhất là trong các trường hợp tải thay đổi (đóng ngắt đầu ra). Nếu cần phải đấu thêm tụ điện ngoài.

[f] Trong đa số các trường hợp, nối đất đầu âm của tất cả các nguồn 24VDC là một trong những cách chống nhiễu tốt nhất.

[g] Điện trở cho phép dòng điện rò chạy qua để chống hiện tượng tích điện tĩnh (thường có giá trị khoảng 1MΩ). Tụ điện chống các nhiễu hài bậc cao (thường có giá trị khoảng 4700 pF).

[h] Nối đầu đấu mát của PLC vào điểm nối đất gần nhất để chống nhiễu. Tất cả các đầu đấu mát trong một hệ thống nên được đấu vào cùng một điểm. Tốt nhất nên sử dụng dây 14 AWG hay dây 1.5 mm▒.

Chỉ sử dụng nguồn cung cấp 24VDC có cách điện tốt với lưới điện xoay chiều cũng như với các nguồn điện khác.

Một số cách đấu nối nhằm bảo vệ tốt các đầu ra của PLC:

- *Bảo vệ các đầu ra 24V một chiều (Transistors)*

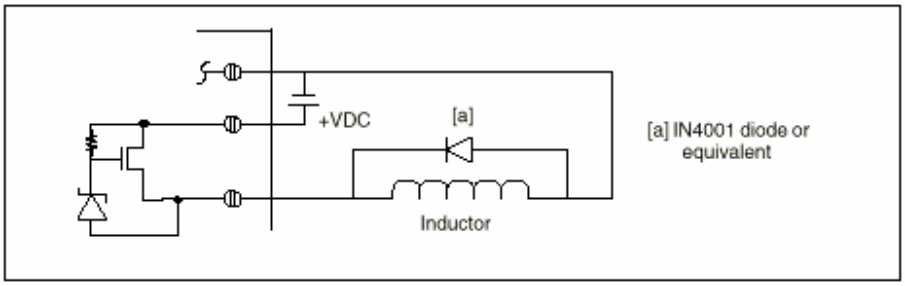


Figure 2-12 Diode Suppression for DC Transistor Outputs

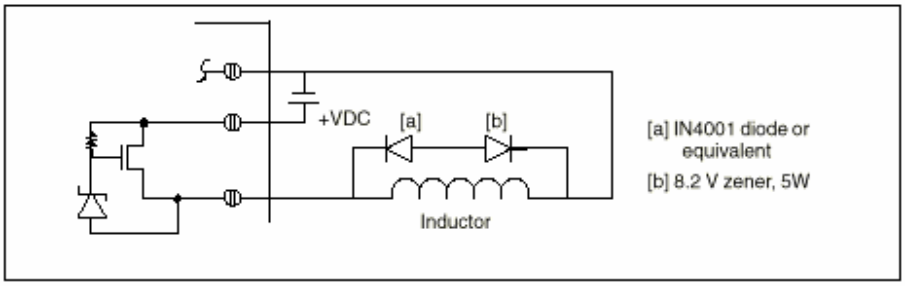


Figure 2-13 Zener Diode Suppression for DC Transistor Outputs

- *Bảo vệ rơ le đóng ngắt dòng điện một chiều*

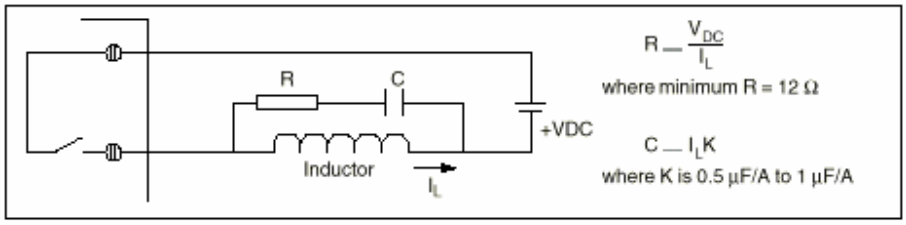


Figure 2-14 Resistor/Capacitor Network on Relay-Driven DC Load

- *Bảo vệ rơ le đóng ngắt dòng điện xoay chiều*

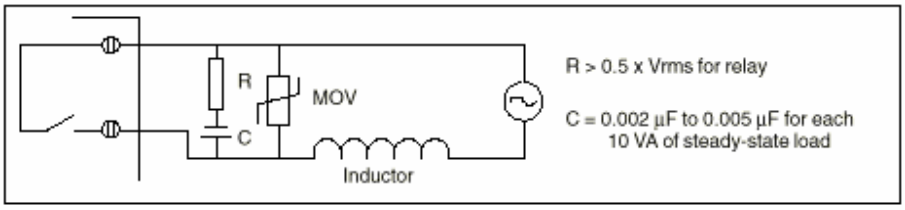


Figure 2-15 AC Load with Network across Relay

3. BẮT ĐẦU LẬP TRÌNH CHO S7-200

Trong chương một, chúng ta đã biết về một hệ thống với S7-200 (Hình 1-2). Một hệ thống như vậy gồm 03 phần:

- S7-200 PLC
- Máy vi tính (PC) hay Thiết bị lập trình được cài đặt phần mềm STEP 7 - Micro / Dos, Win 16 hoặc Win 32.
- Cáp nối chuyên dụng.

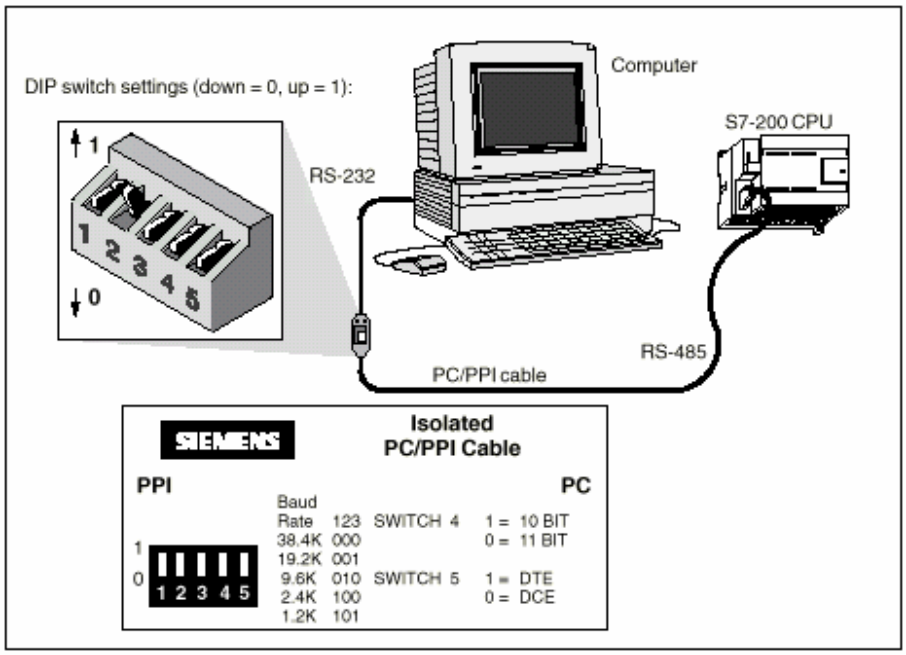
Chúng ta đã điểm qua các CPU S7-200 thông dụng nhất: CPU 212, 214, 215, 216, 221, 222, 224.

Xét hai phần mềm mới nhất, phần mềm STEP 7 - MicroWin 2.1 (16 bits) chỉ lập trình được cho các CPU 212, 214, 215, 216. Từ đây chỉ nói về phần mềm lập trình STEP 7 - MicroWin 3.0 chạy trên môi trường 32 bits, có nhiều tính năng mạnh và lập trình được cho tất cả các loại CPU S7-200 đến thời điểm này. Yêu cầu cấu hình máy vi tính:

- 586 trở lên, 16MB RAM. Tối thiểu là 486 với 8MB RAM.
- Hệ điều hành 32 bits: Windows 85, Windows 98 hoặc Windows NT 4.0.
- Màn hình VGA trở lên.
- Ít nhất 50MB trống trên đĩa cứng.
- Nên có chuột.
- Cáp nối PC/PPI (cắm vào cổng COM1 hay COM 2) hoặc CP Card.

Cài đặt phần mềm đơn giản với lệnh **Run...** ***[drive:]\setup*** (drive chỉ ổ đĩa mềm hoặc CD ROM). Chú ý chọn ngôn ngữ sử dụng (Anh, Đức, Pháp, Tây ban nha hay Ý).

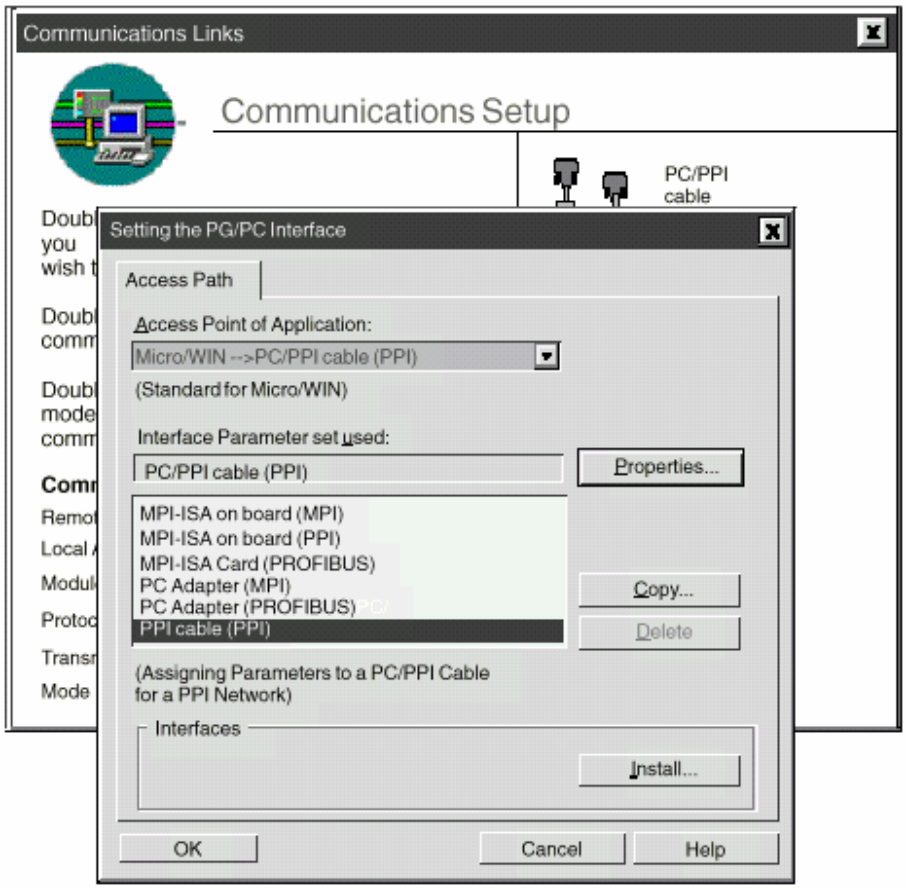
Nối CPU S7-200 với PC bằng cáp PC/PPI



Chọn tốc độ truyền (baud rate) tương ứng với PLC và PC (thông thường 9.6K) bằng DIP switch. Nếu có vị trí số 5 thì chọn 11 BIT và DCE.

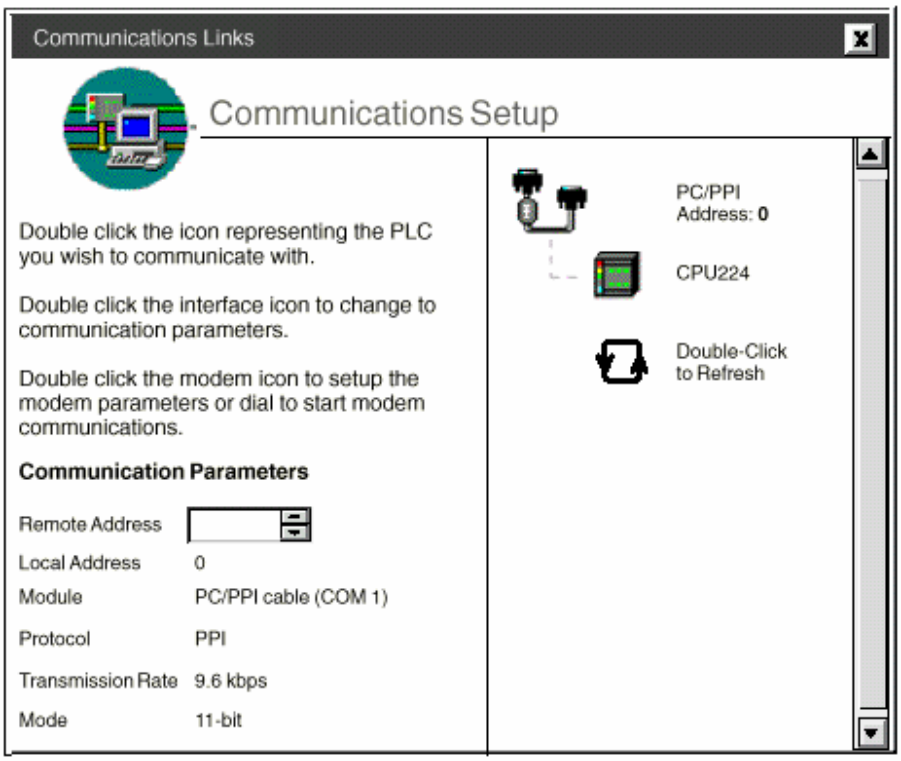
Đặt cấu hình cho cáp PC/PPI

Trong cửa sổ STEP 7 – MicroWin 32, nhắp chuột lên biểu tượng **Communications** hoặc chọn Menu **View > Communications**. Trên hộp đối thoại xuất hiện (**Communications Setup**), nhắp đúp lên biểu tượng **PC/PPI Cable**. Xuất hiện hộp thoại **Setting the PG/PC Interface**, chọn nút **Properties** và kiểm tra các tham số.

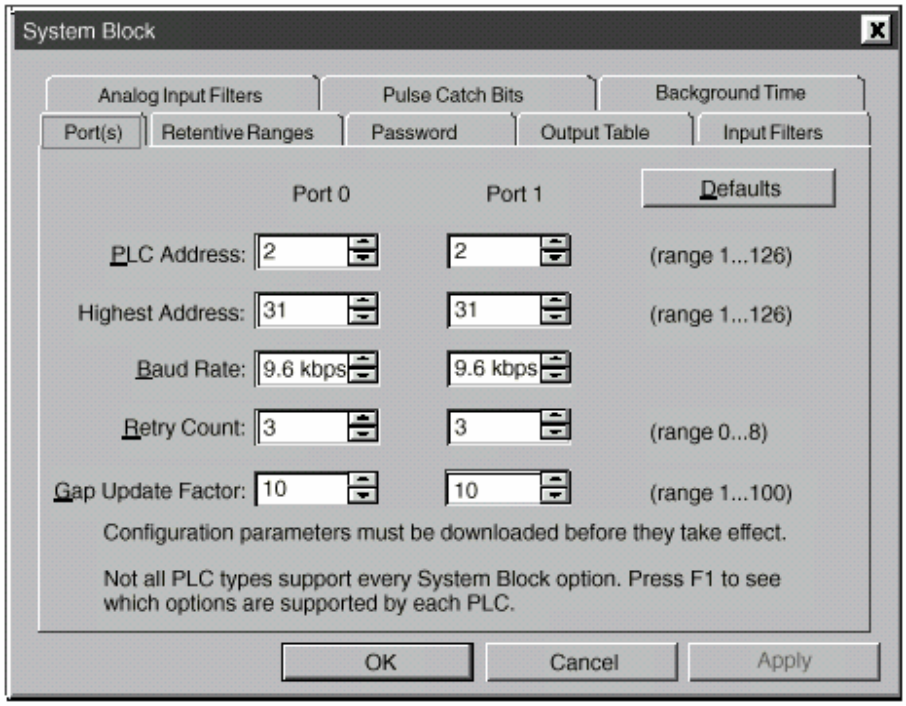


Kết nối với CPU S7-200

Trong cửa sổ STEP 7 - MicroWin 32, nhắp chuột lên biểu tượng **Communications** hoặc chọn Menu **View > Communications**. Trên hộp đối thoại xuất hiện (**Communications Setup**), nhắp đúp lên biểu tượng **Refresh**. CPU đang được kết nối (và được cấp nguồn) sẽ xuất hiện như một biểu tượng. Có thể nhắp đúp lên biểu tượng này để kiểm tra các thông số của PLC tương ứng.



Đặt cấu hình truyền thông cho CPU S7-200



Trong cửa sổ STEP 7 – MicroWin 32, nhắp chuột lên biểu tượng **System Block** hoặc chọn Menu **View > System Block**. Trên hộp đối thoại xuất hiện (**System Block**), chọn trang **Port(s)** để xem và thay đổi các tham số truyền thông.

4. CƠ BẢN TRONG LẬP TRÌNH VỚI S7-200

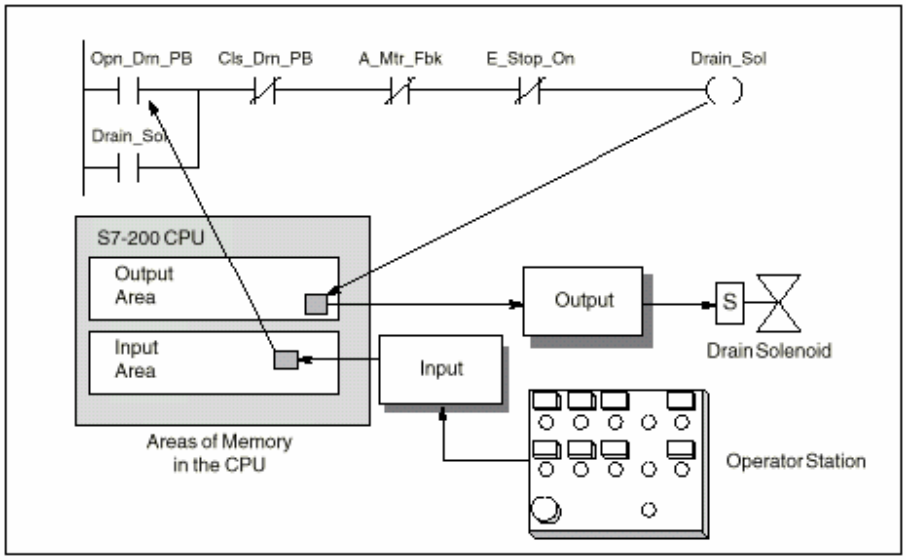
4.1 Tổng quát

Khi tiến hành thiết kế một hệ thống sử dụng PLC S7-200, người thiết kế nhất thiết phải làm quen với những đặc trưng cơ bản nhất của CPU sắp dùng. Thứ hai là phải tuân thủ các nguyên tắc của đơn vị vận hành cũng như của điều kiện thực tế. Không xem như điều bắt buộc nhưng ở đây chúng tôi xin đưa ra một số khuyến cáo chung nhất cho người thiết kế với S7-200.

- Phân chia hệ thống phải thiết kế thành những thành phần nhỏ nhất có mức độ độc lập nhất định. Tiến hành thiết kế từng phần nhỏ một.
- Với mỗi thành phần độc lập, xác định số đầu vào ra; mô tả cách thức hoạt động (đầu ra theo đầu vào); trạng thái "bền" của mỗi bộ phận chấp hành gắn với đầu ra (trạng thái lúc hệ thống phải thiết kế đã đấu nối nhưng chưa hoạt động của các mô tơ, van, . . .); mô tả hình thức vận hành (nút bấm, đèn hiệu, . . .); cuối cùng là cách giao tiếp với các thành phần khác.
- Thiết kế bộ phận an toàn: Ở đây nói đến vấn đề đấu nối phần cứng nhằm đảm bảo an toàn trong những trường hợp bất thường, đặc biệt phải lường trước các trường hợp có thể gây nguy hiểm cho tính mạng con người hay gây thiệt hại lớn về vật chất. Thông thường đây là bộ phận điện cơ hoạt động độc lập với PLC. Trước hết phải thống kê những bộ phận chấp hành nào có thể gây mất an toàn, trong những trường hợp nào. Cần chú ý những trường hợp mất và có điện lại, trường hợp CPU bị treo, bị lỗi. PLC nên có thông tin từ bộ phận an toàn này và tính phương pháp xử lý thích hợp.
- Thiết kế giao diện vận hành: vị trí, cách lắp đặt các nút bấm, đèn hiệu, . . . bố trí hợp lý trên toàn cục hệ thống và kết nối hợp lý với PLC.
- Hoàn thành bản vẽ thiết kế: phân bố cơ học của các chi tiết; sơ đồ đấu nối điện.
- Đặt tên cho các đầu vào ra cũng như các địa chỉ trung gian và tạo thành một bảng.

Mô hình phương thức hoạt động của một chương trình trên PLC khá đơn giản: CPU đọc trạng thái các đầu vào; Chương trình xử lý, cập nhật các thông số theo các đầu vào và thuật toán lô gic định sẵn; CPU xuất tín hiệu ra các đầu ra theo kết quả tạo ra bởi chương trình.

Sau đây là minh họa một ví dụ chương trình đơn giản:



4.2 Ngôn ngữ lập trình

Sẽ không chính xác nếu nói rằng có 03 ngôn ngữ lập trình thông dụng cho PLC, nhưng ở đây chúng ta tạm dùng chữ "ngôn ngữ" để chỉ môi trường (editor) lập trình cho PLC. 03 ngôn ngữ thông dụng đó là:

Statement List (STL)
Ladder Logic (LAD)
Function Block Diagram (FBD)

Với S7-200, mỗi ngôn ngữ có thể sử dụng tập lệnh SIMATIC hay tập lệnh theo chuẩn IEC 1131-3, riêng STL chỉ có thể sử dụng tập lệnh SIMATIC. Chúng ta sẽ đề cập đến vấn đề này sau, trước hết nói về các "ngôn ngữ":

Statement List (STL)

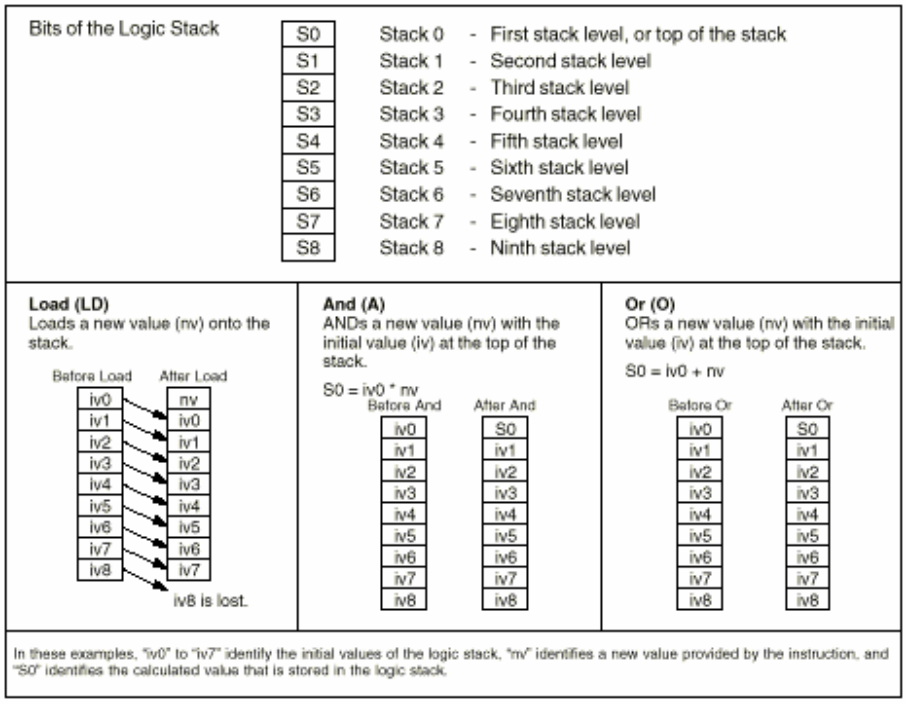
Sau đây là một ví dụ nhỏ: một chương trình S7-200 viết bằng STL.

STL

```
NETWORK
LD    I0.0
LD    I0.1
LD    I2.0
A     I2.1
OLD
ALD
=     Q5.0
```

STL cho phép tạo chương trình bằng cách viết từng câu lệnh, khác với hai ngôn ngữ kia là dạng đồ họa. Chính vì thế trong STL có thể viết những chương trình mà trong hai ngôn ngữ còn lại không viết được, bởi vì nó sát với ngôn ngữ máy hơn, không bị giới hạn bởi các qui tắc đồ họa. STL thường dành cho các lập trình viên giàu kinh nghiệm.

Trong ví dụ trên chúng ta cũng dễ dàng nhận thấy STL có nhiều nét tương tự ngôn ngữ lập trình Assembler. Một khái niệm rất quan trọng trong STL là Ngăn xếp (Stack), khái niệm này không có trong LAD và FBD. Ngăn xếp trong STL về kích thước nhỏ hơn nhiều so với khái niệm ngăn xếp trong Assembler, chỉ bao gồm 09 bits. Tuy nhiên nó lại đóng vai trò lớn hơn, ảnh hưởng tới sự thực hiện của hầu hết các lệnh và các lệnh cũng luôn tác động tới nội dung ngăn xếp. Ta sẽ xét kỹ về ngăn xếp khi đi vào các lệnh cụ thể trong STL.

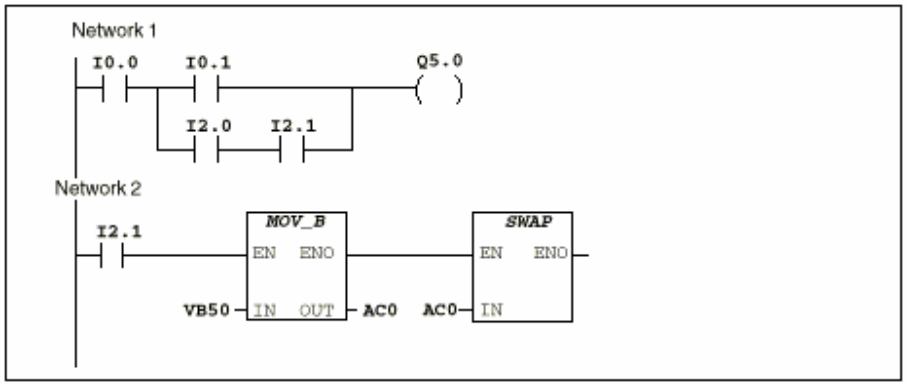


Như trên đã nêu, STL thường dành cho các lập trình viên giàu kinh nghiệm; STL có thể giải quyết được một số vấn đề không thể giải quyết dễ dàng trong LAD và FBD; STL chỉ có thể sử dụng với tập lệnh SIMATIC; Mọi chương trình viết bằng LAD hay

FBD đều có thể chuyển về xem và sửa trong STL nhưng không phải tất cả những chương trình viết trong STL đều có thể xem bằng LAD hoặc FBD.

Ladder Logic (LAD)

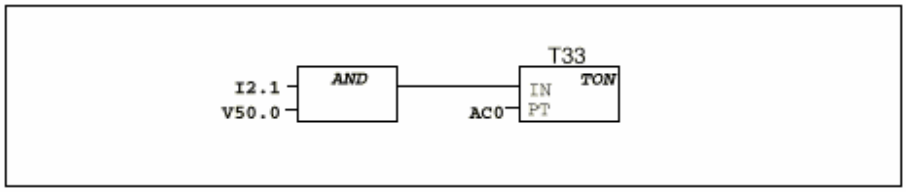
Một chương trình viết trong LAD rất giống với một sơ đồ điện, chinh vì thế mà đây là ngôn ngữ được rất nhiều người lựa chọn khi lập trình cho PLC nói chung. Chương trình thường được chia thành nhiều phần nhỏ, rất dễ hiểu và tương đối độc lập gọi là "rung" hay "network". Những thành phần cơ bản của một chương trình trong LAD là các tiếp điểm (contacts) – đại diện cho các đầu vào như nút bấm, tiếp điểm, điều kiện, . . . các cuộn dây (coils) – đại diện cho các đầu ra như đèn, van, cuộn hút, . . và các hộp (box) –



đặc trưng cho các phép tính, các bộ định thời, các bộ đếm, . . .

Những lý do chính để LAD được yêu thích là: dễ hiểu cho người mới bắt đầu; dễ sử dụng và thông dụng trên toàn thế giới; bao gồm tập lệnh SIMATIC và cả IEC 1131-3; dễ dàng chuyển sang dạng STL.

Function Block Diagram (FBD)



Ví dụ chương trình trong FBD cho thấy nó rất giống với một sơ đồ mạch điện tử sử dụng kỹ thuật số. Đó cũng chính là một ưu điểm của FBD, ngoài ra FBD bao gồm cả tập lệnh SIMATIC và IEC 1131-3 và dễ dàng chuyển sang dạng STL.

4.3 Phân biệt SIMATIC với IEC 1131-3

Tập lệnh SIMATIC được thiết kế dành cho S7-200 PLC. Tập lệnh này có vẻ riêng và hoạt động cũng có hơi khác so với các tập lệnh dành cho các loại PLC khác. Tuy nhiên hầu hết các loại PLC trên thế giới đều sử dụng những tập lệnh có rất nhiều nét tương đồng như tập lệnh này, với đôi nét khác biệt nhỏ giữa các nhà sản xuất PLC khác nhau. Đối với S7-200, các lệnh SIMATIC là tối ưu về mặt thời gian (thực hiện nhanh nhất). Ngoài ra tập lệnh SIMATIC sử dụng được trong cả ba ngôn ngữ STL, LAD và FBD.

Tập lệnh IEC 1131-3, đối lại, tuân thủ theo đúng chuẩn qui định bởi Ủy ban Kỹ thuật Điện Quốc tế (**I**nternational **E**lectrotechnical **C**ommission). Ủy ban này là một tổ chức có hoạt động rộng rãi cũng như có uy tín cao trên thế giới. Trong vài năm trở lại đây, cùng với sự phát triển mạnh mẽ của PLC, IEC cố gắng đưa ra một chuẩn chung nhằm thống nhất các nhà sản xuất PLC khắp nơi trên toàn cầu, để xây dựng một tập lệnh có hình thức cũng như cách hoạt động giống nhau cho mọi loại PLC, tạo dễ dàng cho người sử dụng.

Như vậy, tập lệnh IEC 1131-3 bị giới hạn trong số các lệnh chung nhất của các nhà sản xuất PLC khác nhau trên thế giới. Nhiều lệnh bình thường trong SIMATIC không còn là lệnh chuẩn trong hệ IEC 1131-3. Tất nhiên, người sử dụng vẫn có thể dùng những lệnh này trong IEC 1131-3 như các lệnh 'ngoại chuẩn', nhưng khi đó chương trình không còn hoàn toàn tương thích với chuẩn IEC 1131-3 nữa.

Một số lệnh trong IEC 1131-3 chấp nhận nhiều dạng dữ liệu. Ví dụ lệnh cộng số học trong SIMATIC có nhiều kiểu lệnh: ADD_I để cộng các số nguyên, ADD_R dành cho các số thực; Trong khi đó chỉ có một lệnh cộng ADD trong IEC 1131-3, lệnh này tự động kiểm tra dạng dữ liệu của các toán hạng và biên dịch thành lệnh thích hợp cho CPU. Điều này, cũng được gọi là "overloading", tiết kiệm thời gian quí giá cho người lập trình.

Các lỗi cú pháp ít hơn trong IEC 1131-3 vì dạng dữ liệu được tự động kiểm tra.

Tóm lại với tập lệnh theo chuẩn IEC 1131-3, người sử dụng dễ dàng hơn trong việc làm quen với PLC nói chung. Số lệnh được sử dụng cũng ít hơn, tuy nhiên các lệnh SIMATIC vẫn có thể được sử dụng. Nhiều lệnh khác với những lệnh tương ứng trong SIMATIC như các bộ định thời, bộ đếm, các lệnh nhân, chia, . . . Các lệnh trong IEC 1131-3 có thể có thời gian thực hiện lâu hơn. Các lệnh này chỉ có trong LAD và FBD

(không áp dụng được trong STL). IEC 1131-3 chỉ định rằng phải định nghĩa dạng dữ liệu cho các biến và cung cấp khả năng kiểm tra tính hợp lệ của các biến.

Trong nội dung tài liệu này chúng ta sẽ không đi sâu hơn về vấn đề đang nêu mà chỉ điểm qua một số khái niệm cơ bản. Trước hết là những dạng dữ liệu cơ bản:

Elementary Data Types	Description	Data Range
BOOL	Boolean	0 to 1
BYTE	Unsigned byte	0 to 255
WORD	Unsigned integer	0 to 65,535
INT	Signed integer	-32768 to +32767
DWORD	Unsigned double integer	0 to $2^{32} - 1$
DINT	Signed double integer	-2^{31} to $+2^{31} - 1$
REAL	IEEE 32-bit floating point	-10^{38} to $+10^{38}$

Có 03 mức kiểm tra tính hợp lệ của dữ liệu: kiểm tra chặt chẽ (strong data type checking), kiểm tra đơn giản (simple data type checking) hoặc không kiểm tra (no data type checking). Trong IEC 1131-3 áp dụng mức kiểm tra chặt chẽ còn trong SIMATIC chỉ kiểm tra đơn giản. Kiểm tra chặt chẽ nghĩa là dạng dữ liệu phải tuyệt đối phù hợp, thường thì mỗi lệnh yêu cầu đúng một loại dữ liệu nào đó và điều này phải được đáp ứng (tất nhiên không kể trường hợp các lệnh "overloading" như đã nêu ở trên). Trong khi đó kiểm tra đơn giản chỉ kiểm tra dung lượng bộ nhớ của biến (số bit mà biến đó chiếm), ví dụ biến dạng WORD (không dấu) và dạng INT (có dấu) không bị phân biệt vì đều chiếm 16 bit trong bộ nhớ. Lưu ý trong kiểm tra đơn giản, dạng REAL vẫn được phân biệt riêng dù cũng chiếm 32 bit như các dạng DWORD và DINT. Không kiểm tra dạng dữ liệu áp dụng cho các biến chung (global) trong SIMATIC, ví dụ VD100 chiếm 32 bit có thể được hiểu như DWORD, DINT hay REAL.

Sau đây là các dạng dữ liệu tổng hợp:

Complex Data Types	Description	Address Range
TON[1]	On-Delay Timer	1 ms T32, T96 10 ms T33 to T36, T97 to T100 100 ms T37 to T63, T101 to T255
TOF	Off-Delay Timer	1 ms T32, T96 10 ms T33 to T36, T97 to T100 100 ms T37 to T63, T101 to T255
TP	Pulse Timer (See Note 1)	1 ms T32, T96 10 ms T33 to T36, T97 to T100 100 ms T37 to T63, T101 to T255
CTU	Up Counter	0 to 255
CTD	Down Counter	0 to 255
CTUD	Up/Down Counter	0 to 255
SR	Set Dominant Bistable	--
RS	Reset Dominant Bistable	--
[1] The pulse timer function block uses TON timers to perform the pulse operation. This will reduce the total number of available TON timers.		

Việc kiểm tra tính hợp lệ của dữ liệu hay không kiểm tra đóng vai trò rất quan trọng. Ví dụ trong các lệnh so sánh số nguyên (>I, <I), nếu số dạng WORD được hiểu là số dạng INT thì PLC có thể cho rằng 40000 nhỏ hơn 1.

Do việc kiểm tra tính hợp lệ của dữ liệu trong IEC 1131-3 và SIMATIC khác nhau nên không thể chuyển đổi chương trình giữa hai dạng lệnh này được. Cần phải lựa chọn một tập lệnh duy nhất để sử dụng ngay từ đầu, khi bắt đầu tạo chương trình.

Như trên có nhắc đến các lệnh "overloading", sau đây là một ví dụ đơn giản về việc kiểm tra dạng dữ liệu cho những lệnh này: ta thực hiện lệnh cộng ADD hai toán hạng IN1 (dạng INT) và IN2 (dạng WORD), lưu kết quả vào OUT (dạng INT). Trong IEC 1131-3 sẽ báo lỗi biên dịch (kiểm tra chặt chẽ) còn với kiểm tra bình thường thì lệnh trên được hiểu là lệnh ADD_I (cộng số nguyên). Khi kiểm tra bình thường (đơn giản), lệnh cộng ADD hai số 40000 và 1 sẽ cho kết quả là một số âm chứ không phải là 40001.

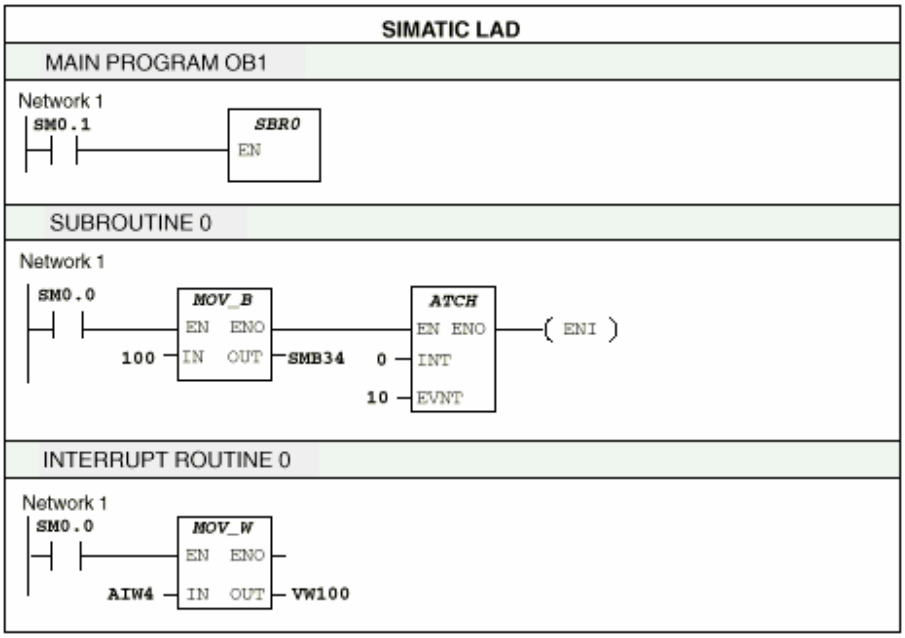
Một điều cũng nên nhắc đến là các lệnh "overloading" sử dụng cách đánh địa chỉ gián tiếp. Do cách đánh địa chỉ gián tiếp không xác định dạng dữ liệu của toán hạng nên lệnh thực hiện tự xác định theo dạng của các toán hạng còn lại. Khi không làm được điều này (toàn địa chỉ gián tiếp hay sử dụng accumulator chẳng hạn) thì sẽ báo lỗi biên dịch.

Điều cuối cùng cần nói đến trong phần này là việc chuyển dạng dữ liệu. Tồn tại các lệnh riêng để chuyển số liệu từ dạng này sang dạng khác, chẳng hạn chuyển số -5 (dạng INT) thành -5.00 (dạng REAL). Một cách chuyển dạng dữ liệu khá thông dụng là bằng lệnh "overloading" MOVE, cho phép chuyển số liệu khác dạng nhưng cùng kích thước (chiếm cùng số bit trong bộ nhớ, ví dụ như INT và WORD, DWORD và DINT).

Ghi chú: Những vấn đề liên quan đến các lệnh cụ thể, hãy xem thêm ở phần giải thích về những lệnh đó (Chương 8 Tập lệnh SIMATIC và Chương 9 Tập lệnh IEC 1131-3)

4.4 Cấu trúc chương trình

Sau đây là một chương trình ví dụ:



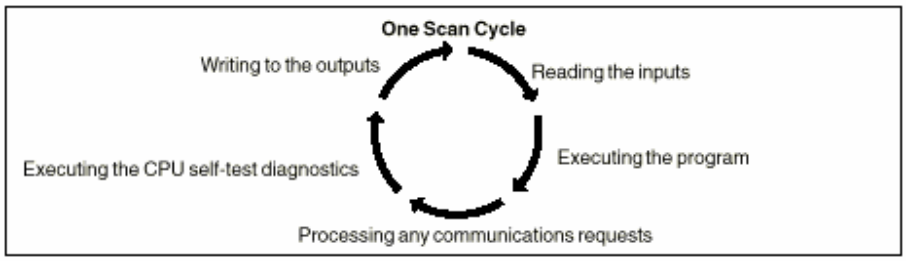
Cấu trúc một chương trình trong PLC khá đơn giản, chương trình được tạo thành từ 03 thành phần cơ bản: một chương trình chính (main program); có thể có một hay nhiều chương trình con (subroutines); các chương trình con xử lý ngắt (interrupt routines) có thể có hoặc không.

- Chương trình chính bao gồm các lệnh điều khiển ứng dụng. Các lệnh này được thực hiện tuần tự một cách liên tục, cứ mỗi vòng quét một lần. Khái niệm vòng quét xem phần tiếp theo (4.5).
- Các chương trình con, có thể có hoặc không tùy yêu cầu, chỉ được thực hiện nếu được gọi đến từ chương trình chính.
- Các chương trình con xử lý ngắt (có thể có hoặc không) được thực hiện khi xảy ra sự kiện gắn với ngắt tương ứng. Sự kiện đó có thể là sự thay đổi mức ở một đầu vào, bộ định thời đếm đủ hay nhận được dữ liệu trên cổng truyền thông, . . .

4.5 Vòng quét

Các lập trình viên trên máy vi tính thường quen với các loại cấu trúc chương trình như: chương trình kiểu dòng lệnh (Assembler, Basic); chương trình kiểu cấu trúc (C, Pascal); chương trình hướng đối tượng (Visual Basic, C, Pascal for Windows). Các kiểu

chương trình này thông thường hoặc kết thúc sau khi thực hiện, hoặc tiếp tục một cấu trúc vòng lặp nào đó chờ tương tác với người sử dụng. Chương trình trong PLC cũng có thể bao gồm các cấu trúc vòng lặp nhưng không phải với mục đích như trên. Chương trình trong PLC nhìn chung giống dạng chương trình kiểu dòng lệnh, ở đó các lệnh được thực thi một cách tuần tự. Tuy nhiên một chương trình trong PLC sẽ được tự động thực hiện một cách tuần hoàn. Cứ một lần chương trình được thực hiện gọi là một vòng quét (SCAN).



Theo hình vẽ chúng ta dễ dàng nhận thấy những công đoạn chính của một vòng quét:

- Đầu tiên là *cập nhật các đầu vào*. Đầu mỗi vòng quét, CPU đọc trạng thái các đầu vào vật lý (các đầu vào rời rạc hiện hữu thực tế trên PLC) và ghi vào "vũng ảnh các đầu vào". Đây là một vùng nhớ, mỗi bit trong vùng này là "ảnh" của một đầu vào, "ảnh" được cập nhật trạng thái từ đầu vào vật lý tương ứng chính ở trong công đoạn này. Về sau trong vòng quét, chương trình hiểu các giá trị đầu vào là các giá trị ảnh này, trừ những lệnh truy cập giá trị "tức khắc" (immediate). Lưu ý, các đầu vào tương tự (analog) chỉ được cập nhật như thế nếu bộ lọc (filter) tương ứng hoạt động. Trong trường hợp ngược lại, chương trình sẽ đọc trực tiếp từ đầu vào tương tự vật lý mỗi khi truy cập. Cụ thể hơn về các đầu vào ra sẽ được nói đến ở chương 6.
- Tiếp theo là *thực hiện chương trình*. Đây là thời gian CPU thực thi các lệnh trong chương trình chính một cách tuần tự từ đầu đến cuối. Chương trình xử lý ngắt được thực hiện không liên quan đến vòng quét mà bất cứ lúc nào xảy ra sự kiện liên quan. Chỉ những lệnh vào ra "tức khắc" mới truy cập đến các đầu vào ra vật lý.
- *Thực hiện các yêu cầu truyền thông* là công đoạn CPU xử lý các thông tin nhận được trên cổng truyền thông.
- *CPU tự kiểm tra*, trong công đoạn này CPU tự kiểm tra các thông số của nó, bộ nhớ chương trình (chỉ trong chế độ RUN) và trạng thái các module nếu có.
- Cuối cùng là *ghi các đầu ra*. CPU ghi giá trị "vùng ảnh các đầu ra" ra các đầu ra vật lý. Vùng ảnh này được cập nhật theo chương trình trong quá trình thực hiện chương trình. Khi CPU chuyển từ chế độ RUN sang chế độ STOP, các đầu ra có thể có giá trị như trong "bảng ra", hay giữ nguyên giá trị (xem chương 6). Thông thường mặc định là các đầu ra trở về "0". Riêng các đầu ra tương tự giữ nguyên giá trị được cập nhật sau cùng.

Nếu có sử dụng ngắt, các chương trình xử lý ngắt được lưu như một phần của chương trình trong bộ nhớ. Tuy nhiên chúng không được thực hiện như một phần của vòng quét bình thường. Chúng được thực hiện khi sự kiện tương ứng xảy ra, bất kỳ lúc nào trong vòng quét, theo nguyên tắc ngắt đến trước được xử lý trước, tất nhiên có tính đến mức độ ưu tiên của các loại ngắt khác nhau.

Như trên đã nêu, trong quá trình thực hiện, chương trình truy cập đến các đầu vào và đầu ra thông qua vùng ảnh của chúng. Vùng ảnh các đầu vào được cập nhật từ các đầu vào vật lý một lần trong một vòng quét, ngay ở đầu vòng quét. Vùng ảnh các đầu ra cũng cập nhật ra các đầu ra vật lý cuối mỗi vòng quét. Nguyên tắc này đảm bảo sự đồng bộ cũng như tính ổn định, cân bằng cho hệ thống; quá trình thực hiện chương trình nhanh hơn; khả năng linh động cho phép truy nhập các đầu vào ra chung như tập hợp các bit, byte hay từ đơn, từ kép.

Các lệnh vào ra trực tiếp (tức khắc) cho phép khai thác trạng thái các đầu vào vật lý cũng như xuất ra các đầu ra vật lý ngay thời điểm thực hiện lệnh, không phụ thuộc và vòng quét. Lệnh đọc đầu vào trực tiếp không ảnh hưởng gì đến vùng ảnh các đầu vào. Bit ảnh đầu ra được cập nhật đồng thời với lệnh xuất trực tiếp ra đầu ra đó.

CPU coi các lệnh đối với các đầu vào ra tương tự như các lệnh vào ra trực tiếp, trừ trường hợp ngoại lệ đầu vào tương tự có bộ lọc hoạt động (sẽ nói kỹ hơn ở chương 6).

4.6 Các chế độ hoạt động

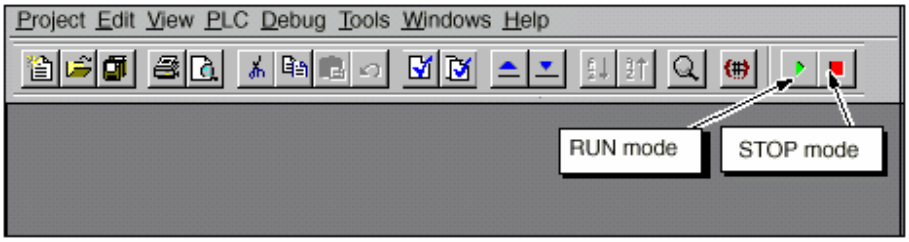
S7-200 PLC có 02 chế độ hoạt động: RUN và STOP.

- Trong chế độ STOP, CPU không thực hiện chương trình. Chúng ta có thể nạp chương trình (download) hay thay đổi cấu hình CPU.
- CPU thực hiện chương trình trong chế độ RUN.

Chế độ hoạt động của CPU được báo hiệu bởi đèn LED phía trước CPU.

Chúng ta có thể chuyển đổi chế độ hoạt động của PLC bằng một trong các cách sau:

- Bật công tắc phía trước mặt CPU. Công tắc này có 03 vị trí: RUN, STOP đại diện cho hai chế độ; vị trí thứ ba TERM không thay đổi chế độ nhưng cho phép có thể chuyển đổi từ phần mềm lập trình. Khi mới bật nguồn, CPU tự động vào chế độ RUN nếu công tắc ở vị trí RUN, ngoài ra CPU tự động vào chế độ STOP.
- Dùng phần mềm STEP 7 - Micro / Win 32 với công tắc nói trên ở vị trí RUN hay TERM.



- Chuyển sang STOP khi chương trình đang chạy bằng lệnh STOP. Lệnh này cho phép ngừng thực hiện chương trình theo ý định lô gic của người lập trình.

4.7 Mật khẩu

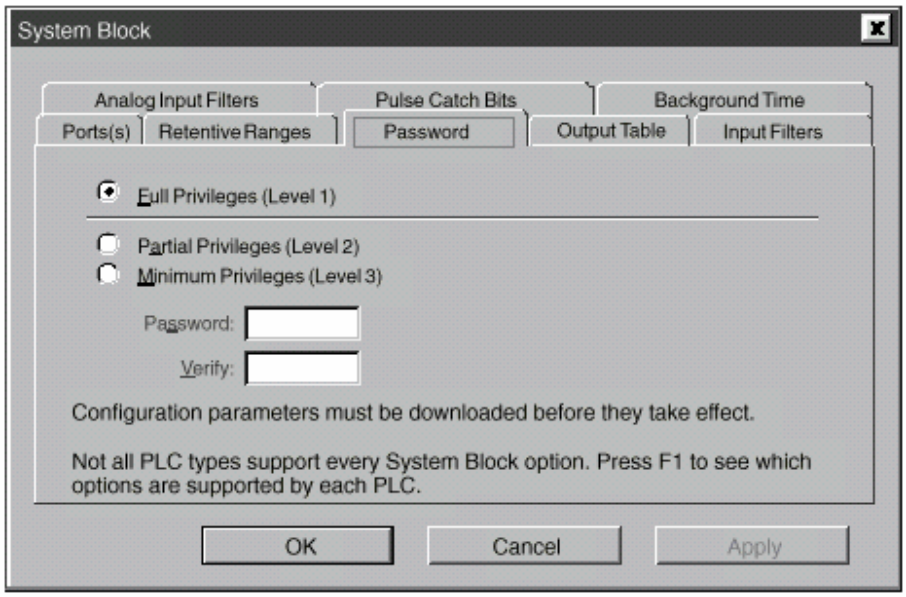
Tất cả các CPU đời S7-200 đều có khả năng bảo vệ và hạn chế truy nhập bằng mật khẩu. Có 03 mức hạn chế, trong đó người sử dụng sẽ được toàn quyền nếu có mật khẩu, nếu không có, người sử dụng sẽ bị hạn chế quyền tùy theo mức được đặt mật khẩu như trong bảng dưới đây:

Task	Level 1	Level 2	Level 3
Read and write user data	Not restricted	Not restricted	Not restricted
Start, stop and restart the CPU			
Read and write the time-of-day clock			
Upload the user program, data, and the configuration			Password required
Download to the CPU		Password required	
Delete the user program, data, and the configuration			
Force data or single/multiple scan			
Copy to the memory cartridge			
Write outputs in STOP mode			

Ta có thể thấy thực tế chỉ có 02 mức bảo vệ, mức 1 chính là mức không hạn chế gì (không có mật khẩu).

Nếu quên mật khẩu, chỉ có cách xóa bộ nhớ của CPU và nạp lại chương trình. Lúc bị xóa bộ nhớ, CPU chuyển về chế độ STOP, cấu hình mặc định như khi mới xuất xưởng trừ địa chỉ CPU, tốc độ truyền thông và đồng hồ thời gian thực. Cần chú ý điều kiện an toàn khi PLC ở trong hệ thống vì tất cả các đầu ra sẽ chuyển về "0". Để xóa, chọn thực đơn **PLC > Clear...** Nếu chương trình có mật khẩu, một hộp thoại sẽ hiện ra hỏi, ta phải gõ vào mật khẩu xóa (*clearplc*). Động tác này không xóa chương trình trong "Cartridge". Vì chương trình được lưu cùng với mật khẩu nên ta cũng phải nạp lại chương trình cho cartridge để xóa mật khẩu cũ.

Đặt mật khẩu bằng cách chọn thực đơn **View > System Block** và chọn trang Password.

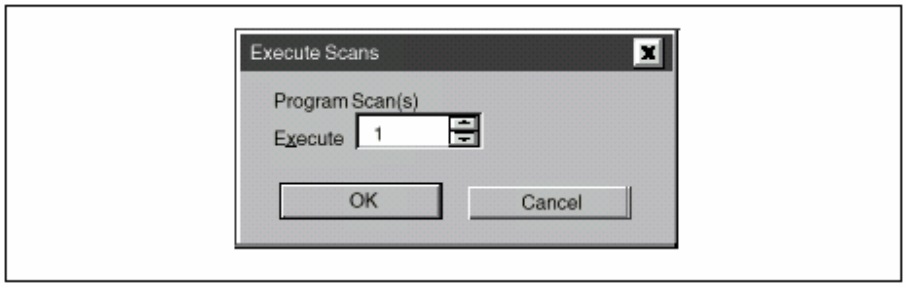


Mật khẩu sẽ có giá trị sau khi được nạp vào PLC.

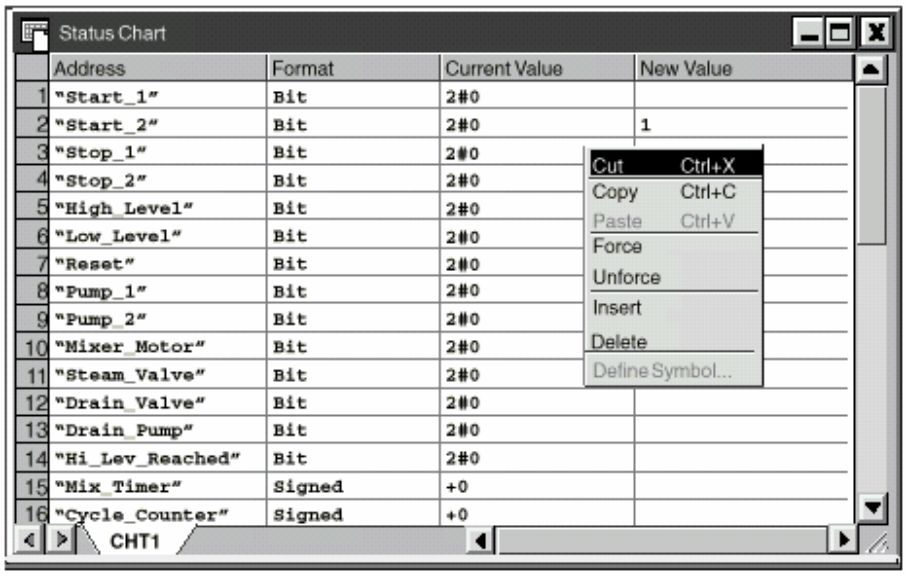
4.8 Gỡ rối

Tài liệu không đi sâu vào vấn đề này, chỉ nêu tên một số phương tiện có sẵn trong môi trường lập trình để giúp người lập trình gỡ rối chương trình (debug).

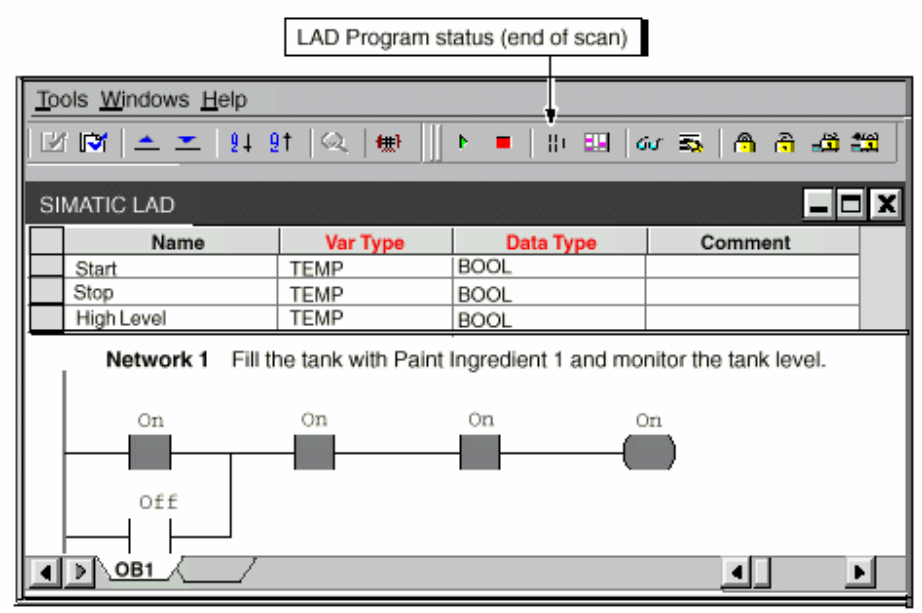
- Chạy chương trình trong một số vòng quét nhất định. Chọn thực đơn **Debug > Multiple Scans** và chọn số vòng quét muốn thực hiện.



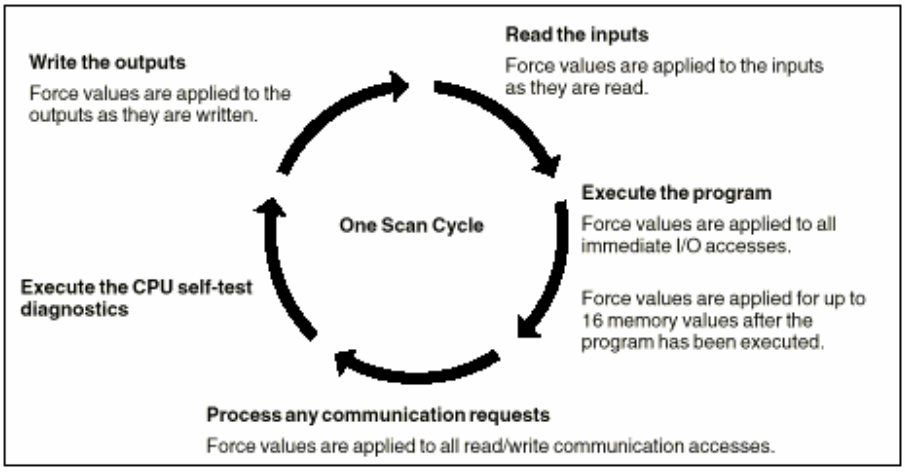
- Sử dụng bảng trạng thái Status Chart. Chọn thực đơn **View > Status Chart** hay nhấn biểu tượng tương ứng. Có thể tạo nhiều bảng khác nhau. Sử dụng phím chuột phải để định cấu hình.

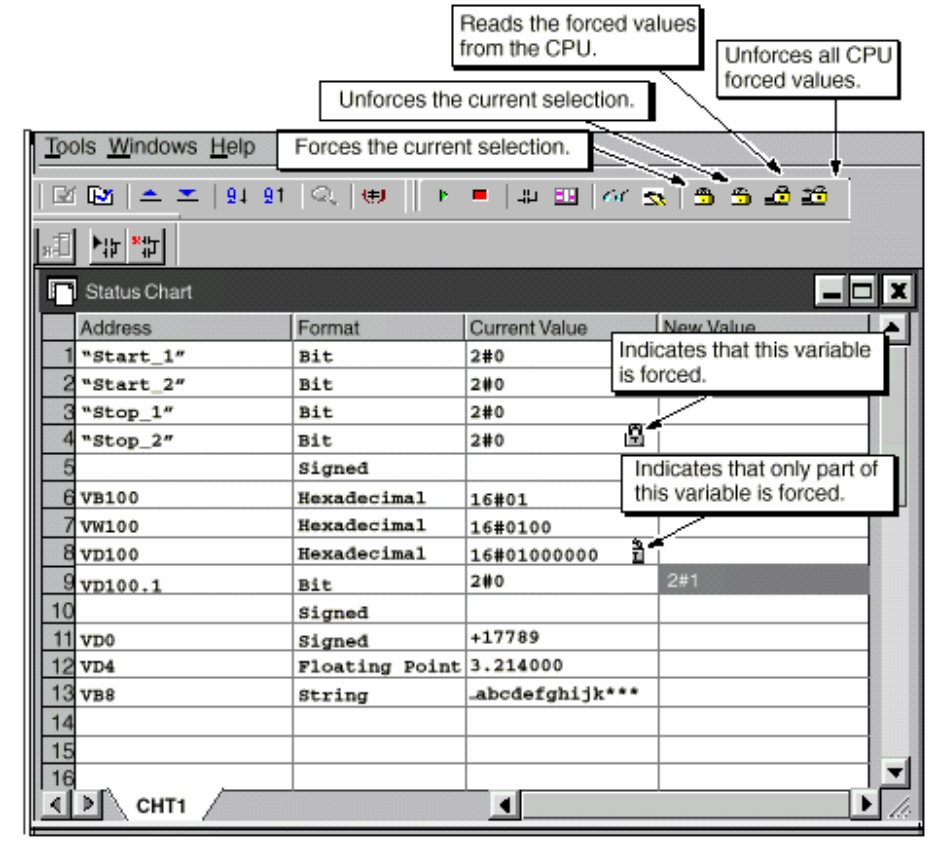


- Hiển thị trạng thái trong cửa sổ chương trình.



- "Force" một số giá trị nào đó. Ta có thể định một số biến nào đó phải bằng một giá trị cố định để kiểm tra các biến liên quan.



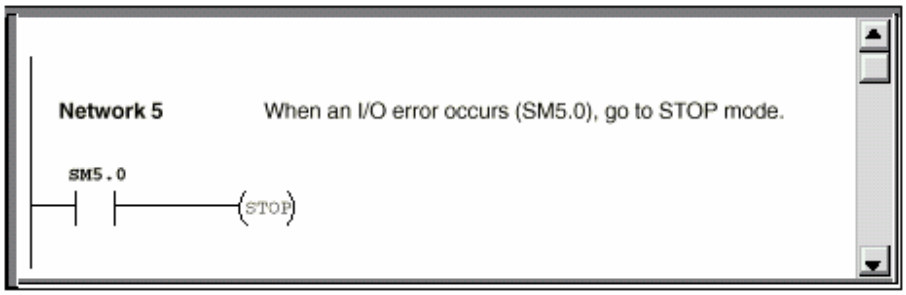


	Address	Format	Current Value	New Value
1	"Start_1"	Bit	2#0	
2	"Start_2"	Bit	2#0	
3	"Stop_1"	Bit	2#0	
4	"Stop_2"	Bit	2#0	
5		Signed		
6	VB100	Hexadecimal	16#01	
7	VW100	Hexadecimal	16#0100	
8	VD100	Hexadecimal	16#01000000	
9	VD100.1	Bit	2#0	2#1
10		Signed		
11	VD0	Signed	+17789	
12	VD4	Floating Point	3.214000	
13	VB8	String	abcdefghijk***	
14				
15				
16				

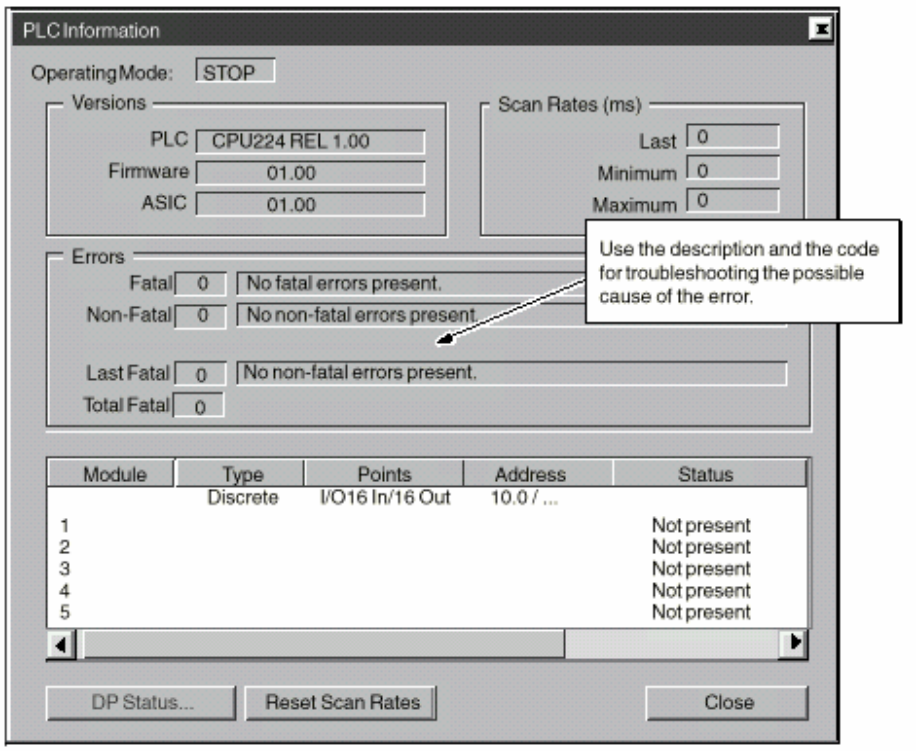
4.9 Thông báo và xử lý lỗi

Phần này chỉ dành cho lập trình viên có kinh nghiệm. Thông thường lỗi được chia thành 02 loại chính: nghiêm trọng và không nghiêm trọng (fatal errors & non-fatal

errors). Lỗi nghiêm trọng gây ngừng chương trình và ta phải tiến hành "Reset" (bằng một trong 03 cách: tắt rồi bật nguồn, chuyển công tắc về STOP rồi bật lên lại, chọn thực đơn **PLC > Power-Up Reset**), lỗi này có thể được thông báo trên đèn LED phía trước CPU. Lỗi không nghiêm trọng bao gồm lỗi lúc chạy chương trình (run-time errors), lỗi lúc biên dịch (program-compile errors) và lỗi do chương trình thực hiện. Lỗi không nghiêm trọng không gây ngừng chương trình, trừ khi được lập trình với lệnh STOP, ví dụ:



Lỗi do chương trình thực hiện là lỗi gây nên bởi lô gic của người lập trình. Ta có thể xử lý các lỗi còn lại với sự trợ giúp của phương tiện lập trình (chọn thực đơn **PLC > Information**) và tra mã lỗi trong phụ lục kèm theo (*B Error Codes*).

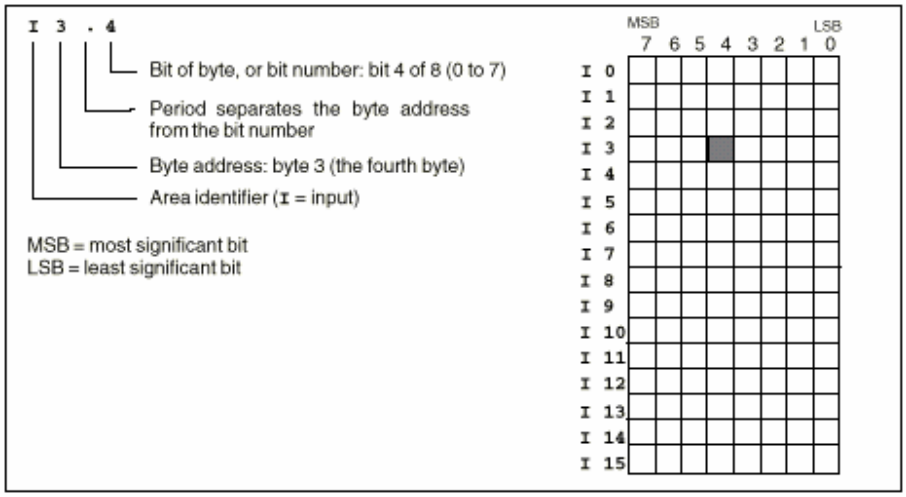


5. BỘ NHỚ DỮ LIỆU VÀ CÁCH ĐỊNH ĐỊA CHỈ

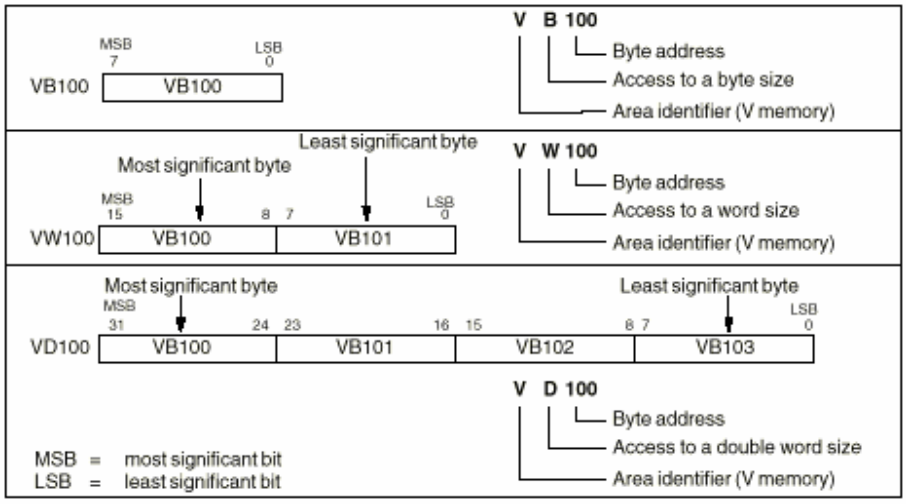
S7-200 PLC quản lý bộ nhớ dữ liệu theo từng vùng riêng biệt nhằm xử lý nhanh hơn và hiệu quả hơn. Đó là các vùng I, Q, V, M, S, SM, L, T, C, HC, AC. Ta sẽ xem xét từng vùng cụ thể ở phần sau.

5.1 Định địa chỉ trực tiếp

Trong các vùng cơ bản I, Q, V, M, S, SM, L ta có thể truy cập đến từng bit, từng byte, từng từ đơn (word) hoặc từng từ kép (double word) dựa trên địa chỉ cơ sở là địa chỉ byte.



Trên đây là một ví dụ cách định địa chỉ một bit: trước hết là tên vùng (I, Q, V, M, S, SM, L), tiếp theo là địa chỉ byte trong vùng đó, cuối cùng sau dấu chấm là địa chỉ bit ở trong byte nói trên (từ 0 đến 7).



Muốn truy cập đến một byte trong một vùng nào đó, trước hết phải định vùng (I, Q, V, M, S, SM, L), tiếp theo là B (đặc trưng cho byte) và địa chỉ byte trong vùng. Địa chỉ một từ đơn hoặc một từ kép cũng bắt đầu bằng tên vùng (I, Q, V, M, S, SM, L), tiếp theo là W (word) hay D (double word) và sau cùng là địa chỉ byte đầu tiên trong từ (byte cao nhất). (Xem các ví dụ phía trên). Tùy theo kích thước ô nhớ được truy cập (dung lượng chiếm trong bộ nhớ) mà con số sử dụng sẽ bị giới hạn, ví dụ với các số nguyên:

Data Size	Unsigned Integer Range		Signed Integer Range	
	Decimal	Hexadecimal	Decimal	Hexadecimal
B (Byte): 8-bit value	0 to 255	0 to FF	-128 to 127	80 to 7F
W (Word): 16-bit value	0 to 65,535	0 to FFFF	-32,768 to 32,767	8000 to 7FFF
D (Double word, Dword): 32-bit value	0 to 4,294,967,295	0 to FFFF FFFF	-2,147,483,648 to 2,147,483,647	8000 0000 to 7FFF FFFF

Riêng giới hạn cho số thực (32 bit), dương từ +1.175495e-38 đến +3.402823e+38, âm từ -1.175495e-38 đến -3.402823e+38.

Đối với các vùng thiết bị (T, C, HC, AC), ta truy cập đến bằng tên vùng và địa chỉ thiết bị.

Sau đây ta xét đến từng vùng cụ thể:

*Vùng ảnh các đầu vào **I***

Như đã nêu, CPU lấy mẫu các đầu vào mỗi vòng quét một lần và lưu giá trị vào vùng ảnh. Sau đó chương trình truy nhập vào vùng ảnh này, đến từng bit, từng byte, từng từ đơn hoặc từng từ kép bằng cách định địa chỉ ô nhớ tương ứng:

Bit	I[*byte address*].[*bit address*]	I0.1
Byte, Word, Double Word	I[*size*][*starting byte address*]	IB4

trong đó: bit address = từ 0 đến 7
byte address = từ 0 đến giới hạn bởi từng loại CPU cụ thể
size = B với byte; W với từ đơn; D với từ kép

*Vùng ảnh các đầu ra **Q***

Chương trình truy xuất các đầu ra thông qua vùng ảnh các đầu ra, vùng ảnh này được ghi ra các đầu ra vật lý mỗi vòng quét một lần ở cuối vòng quét. Chương trình truy xuất các đầu ra có thể như một bit, một byte hay một từ đơn, từ kép:

Bit	Q[*byte address*].[*bit address*]	Q1.2
Byte, Word, Double Word	Q[*size*][*starting byte address*]	QW6

trong đó: bit address = từ 0 đến 7
byte address = từ 0 đến giới hạn bởi từng loại CPU cụ thể
size = B với byte; W với từ đơn; D với từ kép

*Vùng nhớ các biến **V***

Vùng này có thể được sử dụng để lưu các giá trị trung gian, bit, byte, từ đơn hay từ kép:

Bit	V[*byte address*].[*bit address*]	V100.7
Byte, Word, Double Word	V[*size*][*starting byte address*]	VD10

trong đó: bit address = từ 0 đến 7
byte address = từ 0 đến giới hạn bởi từng loại CPU cụ thể
size = B với byte; W với từ đơn; D với từ kép

*Vùng nhớ các bit **M***

Vùng M có tên là vùng nhớ các bit, thực tế chúng ta có thể sử dụng y như vùng V (thường dung lượng vùng M nhỏ hơn):

Bit	M[*byte address*].[*bit address*]	M0.3
Byte, Word, Double Word	M[*size*][*starting byte address*]	MW4

trong đó: bit address = từ 0 đến 7
byte address = từ 0 đến giới hạn bởi từng loại CPU cụ thể
size = B với byte; W với từ đơn; D với từ kép

*Vùng nhớ các rơ le điều khiển tuần tự **S***

Vùng này thường được sử dụng để điều khiển quá trình thực hiện các công đoạn chương trình, cách truy cập giống như các vùng V và M:

Bit	S[*byte address*].[*bit address*]	S0.0
Byte, Word, Double Word	S[*size*][*starting byte address*]	SB4

trong đó: bit address = từ 0 đến 7
byte address = từ 0 đến giới hạn bởi từng loại CPU cụ thể
size = B với byte; W với từ đơn; D với từ kép

*Vùng các bit đặc biệt **SM***

Mỗi ô nhớ trong vùng SM (bit, byte, từ đơn, từ kép) đều có một ý nghĩa gì đó đối với hệ thống. Khi đọc trạng thái ô nhớ từ vùng SM, ta biết thông tin về PLC và khi ghi dữ liệu vào đó, ta có thể thay đổi tham số, cấu hình của PLC. Cụ thể hơn xem phụ lục (Appendix C). Tuy gọi là các bit đặc biệt nhưng ta có thể truy nhập như bit, cả như byte, từ đơn hay từ kép:

Bit	SM[*byte address*].[*bit address*]	SM0.1
Byte, Word, Double Word	SM[*size*][*starting byte address*]	SMB86

trong đó: bit address = từ 0 đến 7
byte address = từ 0 đến giới hạn bởi từng loại CPU cụ thể
size = B với byte; W với từ đơn; D với từ kép

*Vùng nhớ cục bộ **L***

Về mặt sử dụng, vùng L giống hệt vùng V nhưng chỉ có dung lượng 64 byte. Chú ý trong LAD và FBD, 04 byte cuối được dành cho mục đích riêng nên chỉ còn 60 byte cho chương trình. Trong STL có thể sử dụng cả 64 byte nhưng cũng có khuyến cáo không nên sử dụng 04 byte cuối.

Mặt khác nhau cơ bản giữa vùng L và vùng V rất quan trọng: trong khi các vùng được nêu đến thời điểm này đều có giá trị toàn cục thì vùng L chỉ có giá trị cục bộ (local). Điều đó có nghĩa là chương trình chính có 64 byte vùng nhớ L riêng của mình, mỗi chương trình con cũng có riêng một vùng L với dung lượng 64 byte và mỗi chương trình xử lý ngắt cũng vậy. Nội dung bit nhớ L3.1 trong chương trình chính chẳng có gì chung với bit nhớ L3.1 trong chương trình con số 1. Chương trình con không thể truy cập vùng L của chương trình chính và ngược lại.

Vùng L có giá trị ngẫu nhiên khi chưa được ghi vào, vì vậy phải cẩn thận lúc sử dụng. Có thể sử dụng ô nhớ trong vùng L làm thanh trỏ chứa các địa chỉ gián tiếp nhưng không thể truy nhập vùng L một cách gián tiếp.

Như đã nêu, cách truy nhập vùng L giống truy nhập vùng V:

Bit	L[*byte address*].[*bit address*]	L0.0
Byte, Word, Double Word	L[*size*][*starting byte address*]	LB33

trong đó: bit address = từ 0 đến 7
byte address = từ 0 đến giới hạn bởi từng loại CPU cụ thể
size = B với byte; W với từ đơn; D với từ kép

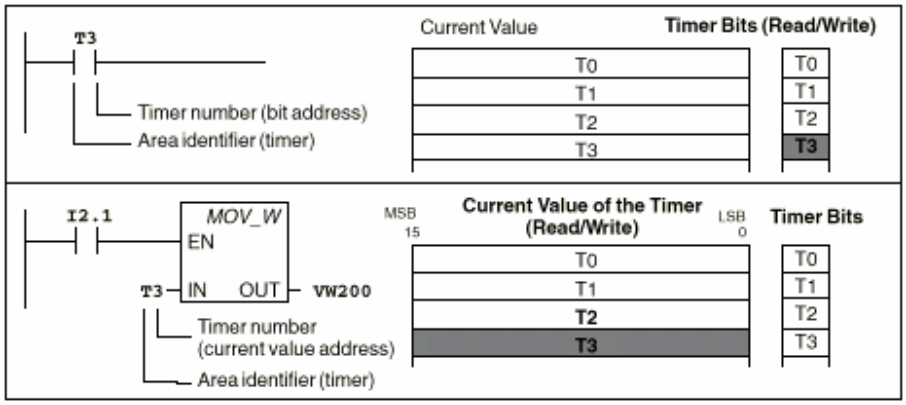
*Vùng các bộ định thời **T***

Trong S7-200, các bộ định thời (timers) được coi là những thiết bị đếm thời gian. PLC S7-200 có 03 loại timer với độ phân giải khác nhau: 1ms, 10 ms và 100ms (thời gian đếm được = số đang đếm * độ phân giải). Mỗi timer đã được xác định cố định một độ phân giải nào đó trong 03 loại nói trên, cách định địa chỉ rất đơn giản:

T[*timer number*] Ví dụ: T24

trong đó: timer number = từ 0 đến giới hạn bởi từng loại CPU cụ thể (thường là 63 hoặc 127).

Một địa chỉ như thế có thể chỉ một *giá trị* 16 bit có dấu là giá trị mà timer đó đang đếm; hoặc chỉ *bit trạng thái* của timer. Chương trình tự phân biệt điều này bằng từng lệnh cụ thể: lệnh có toán hạng kiểu từ đơn sẽ hiểu đó là địa chỉ giá trị timer, ngược lại lệnh có toán hạng kiểu bit sẽ coi đó là địa chỉ bit trạng thái. Xem các ví dụ sau:



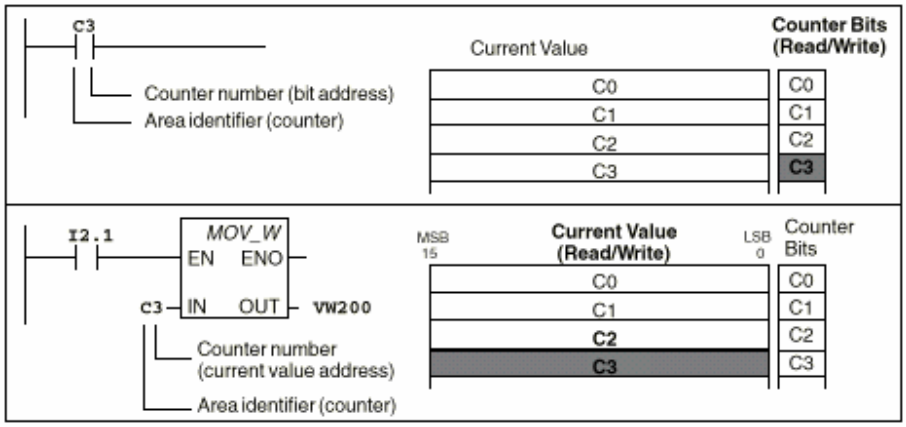
cụ thể hơn chúng ta sẽ nói đến ở phần lệnh (chương 9).

Vùng các bộ đếm ***C***

Các bộ đếm trong S7-200 đếm sự thay đổi đầu vào của chúng từ mức thấp lên mức cao. Chúng có thể đếm lên (tiến), đếm xuống (lùi) hoặc cả đếm tiến lẫn đếm lùi. Cách định địa chỉ một bộ đếm (counter):

C[*counter number*] Ví dụ: C20

trong đó: counter number = từ 0 đến giới hạn bởi từng loại CPU cụ thể (thường là 63 hoặc 127).



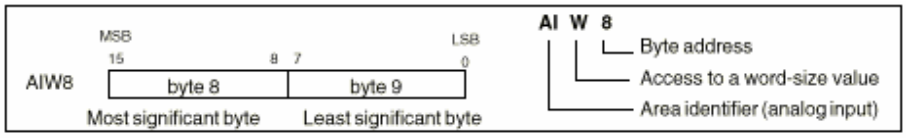
Một địa chỉ như thế có thể chỉ một *giá trị* 16 bit có dấu là giá trị mà counter đó đang đếm; hoặc chỉ *bit trạng thái* của counter. Chương trình tự phân biệt điều này bằng từng lệnh cụ thể: lệnh có toán hạng kiểu từ đơn sẽ hiểu đó là địa chỉ giá trị counter, ngược lại lệnh có toán hạng kiểu bit sẽ coi đó là địa chỉ bit trạng thái. Cụ thể hơn chúng ta sẽ nói đến ở phần lệnh (chương 9).

*Vùng các đầu vào tương tự **AI***

S7-200 chuyển các giá trị tương tự thành những giá trị số 16 bit nên vùng này chỉ được truy nhập đến như những từ đơn:

AIW[*starting byte address*] Ví dụ: AIW4

trong đó: starting byte address = từ 0 đến giới hạn bởi từng loại CPU cụ thể nhưng luôn luôn là số chẵn (0, 2, 4, 6, . . .).



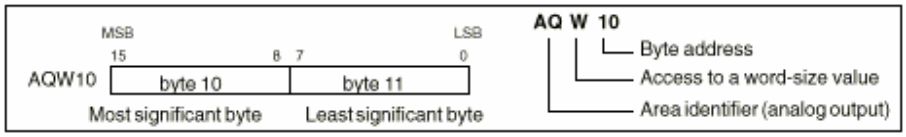
Chú ý đây là các giá trị chỉ đọc (không ghi vào đó được).

*Vùng các đầu ra tương tự **AQ***

S7-200 chuyển những giá trị số 16 bit thành các giá trị ra tương tự nên vùng này cũng chỉ được truy nhập đến như những từ đơn:

AQW[*starting byte address*] Ví dụ: AQW4

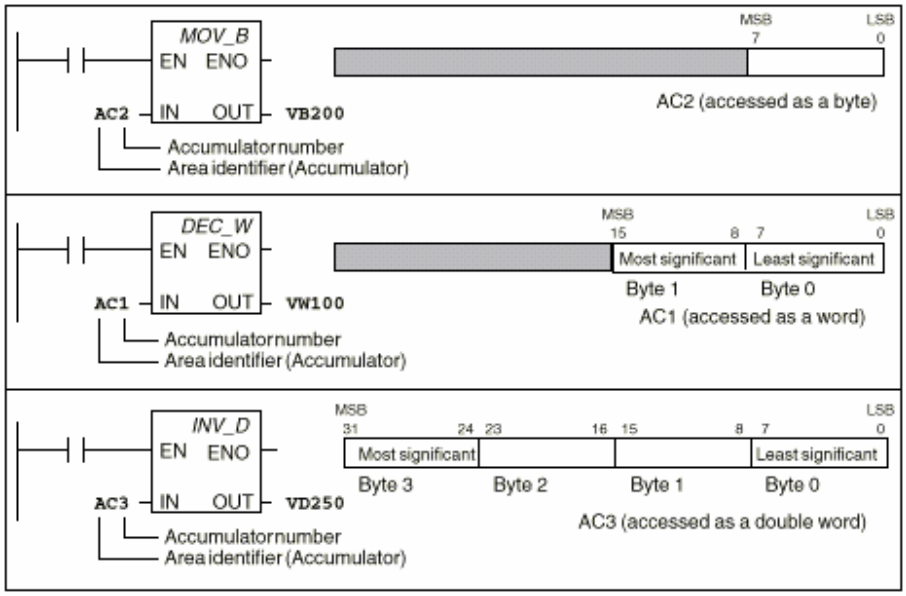
trong đó: starting byte address = từ 0 đến giới hạn bởi từng loại CPU cụ thể nhưng luôn luôn là số chẵn (0, 2, 4, 6, . . .).



Chú ý đây là các giá trị chỉ ghi (không có ý nghĩa đọc từ đó).

*Các accumulator **AC***

S7-200 bao gồm 04 accumulator dung lượng 32 bit: AC0, AC1, AC2 và AC3. Tuy nhiên có thể dùng accumulator để chứa dữ liệu byte, từ đơn hoặc từ kép. Chương trình tự phân biệt điều này bằng lệnh cụ thể (đòi hỏi toán hạng là kiểu byte, từ đơn hay từ kép) như các ví dụ sau:



Các accumulator được sử dụng như những thanh ghi (registers) đọc / ghi đa năng.

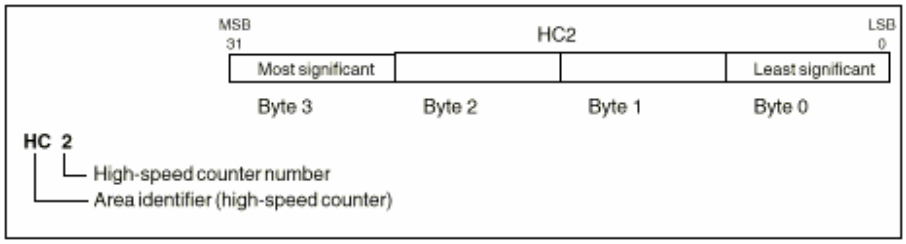
*Các bộ đếm tốc độ cao **HC***

Bộ đếm tốc độ cao trong S7-200 dùng để đếm những đầu vào thay đổi nhanh (tần số cao) độc lập với vòng quét. Địa chỉ bộ đếm tốc độ cao chỉ đến giá trị 32 bit có dấu là con số bộ đếm đang đếm:

HC[*high- speed counter number*] Ví dụ: HC1

trong đó: high-speed counter number = từ 0 đến giới hạn bởi từng loại CPU cụ thể (1,2 hoặc 3).

Con số này là giá trị chỉ đọc, luôn luôn 32 bit.



Các hằng số

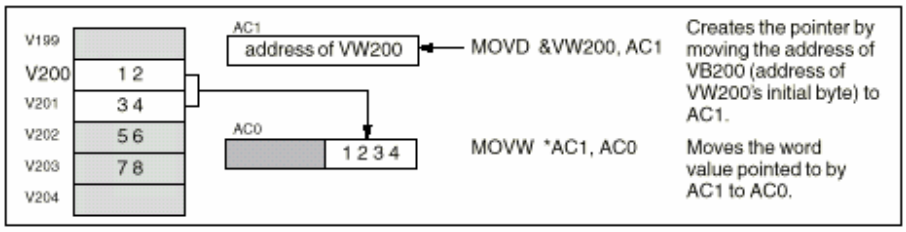
Nhiều lệnh trong S7-200 có thể sử dụng các hằng số dưới các dạng khác nhau, CPU luôn lưu bằng dạng nhị phân. S7-200 CPU không lưu giữ dạng dữ liệu, ví dụ lệnh ADD_I luôn hiểu giá trị đã lưu vào VW100 là số nguyên 16 bit có dấu trong khi lệnh WOR_W lại hiểu đúng giá trị đó trong VW100 là số nguyên 16 bit không dấu. Sau đây là một vài ví dụ về các kiểu hằng số:

- Decimal constant: **20047**
- Hexadecimal constant: **16#4E4F**
- ASCII constant: **'Text goes between single quotes.'**
- Real or floating-point format: **+1.175495E-38 (positive)**
  -1.175495E-38 (negative)
- Binary format **2#1010_0101_1010_0101**

5.2 Định địa chỉ gián tiếp

S7-200 cho phép truy nhập các ô nhớ trong các vùng I, Q, V, M, S, T (chỉ giá trị 16 bit), C (chỉ giá trị 16 bit) một cách gián tiếp, nghĩa là dùng một ô nhớ khác làm thanh trỏ trỏ đến ô nhớ này. Lưu ý không thể truy cập một bit bằng cách gián tiếp.

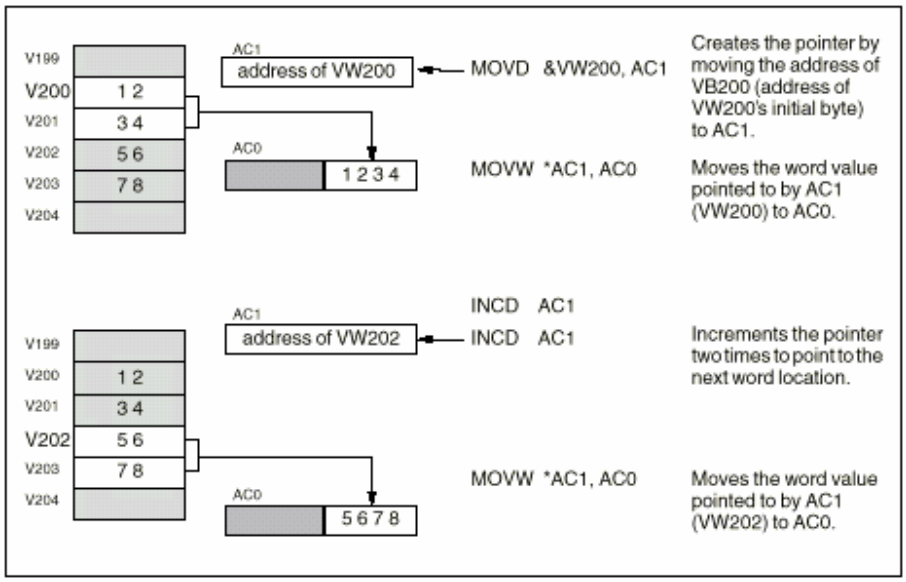
Trong S7-200, thanh trỏ chỉ có thể là một ô nhớ 32 bit (từ kép) trong một trong những vùng V, L hay AC (trừ AC0). Ta có thể tạo thanh trỏ bằng lệnh MOVD với toán tử **&** và sử dụng thanh trỏ bằng toán tử *. Ví dụ:



Chúng ta có thể sử dụng các lệnh số học đơn giản như cộng hoặc tăng 1 dành cho từ kép (ADD_D hoặc INC_D) để thay đổi giá trị thanh trỏ. Tuy nhiên phải đặc biệt chú ý đến kích cỡ dữ liệu mà thanh trỏ đó trỏ đến:

- Nếu một thanh trỏ đang trỏ đến một byte, nó có thể trỏ đến byte kế tiếp bằng cách tăng giá trị nó lên 01 đơn vị.
- Nếu một thanh trỏ đang trỏ đến một từ đơn, nó có thể trỏ đến từ đơn kế tiếp bằng cách tăng giá trị nó lên 02 đơn vị.
- Nếu một thanh trỏ đang trỏ đến một từ kép, nó có thể trỏ đến từ kép kế tiếp bằng cách tăng giá trị nó lên 04 đơn vị.

Ví dụ:

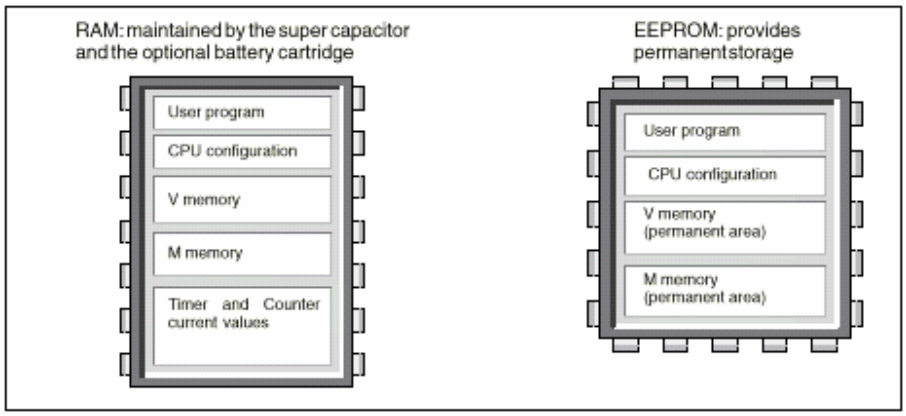


5.3 Bảo toàn dữ liệu

S7-200 cung cấp nhiều khả năng cho phép lưu giữ chương trình, dữ liệu cũng như cấu hình hệ thống trong những trường hợp mất nguồn cung cấp:

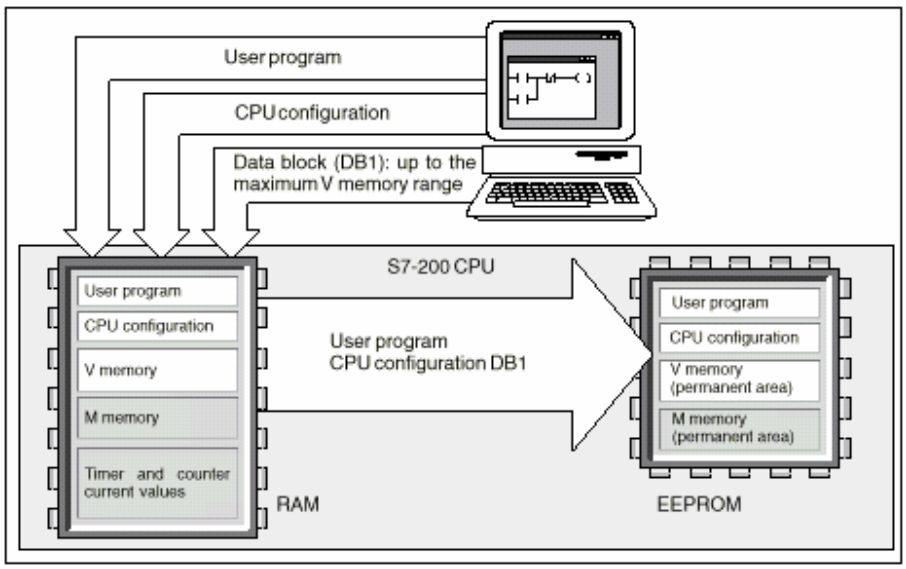
- CPU có bộ nhớ kiểu EEPROM để lưu toàn bộ chương trình, cấu hình và phần dữ liệu quan trọng nhất.

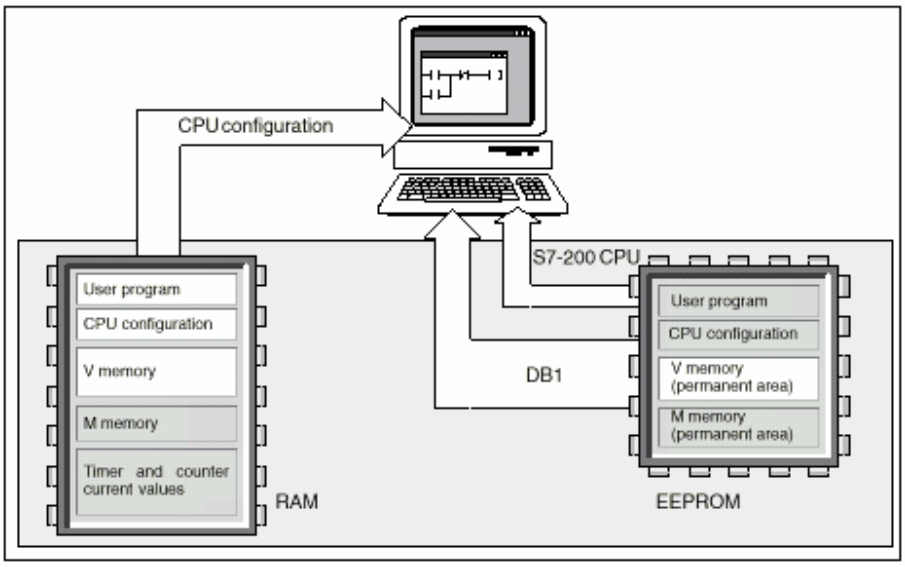
- Bộ nhớ RAM được trang bị "super capacitor" có thể giữ nguyên vẹn thông tin một thời gian dài sau khi mất nguồn nuôi. Tùy loại CPU, thời gian đó có thể kéo dài vài ngày.
- Ta có thể chọn gắn thêm "cartridge" chứa pin để kéo dài thời gian nói trên. Pịn sẽ giữ dữ liệu trong RAM sau khi "super capacitor" cạn.



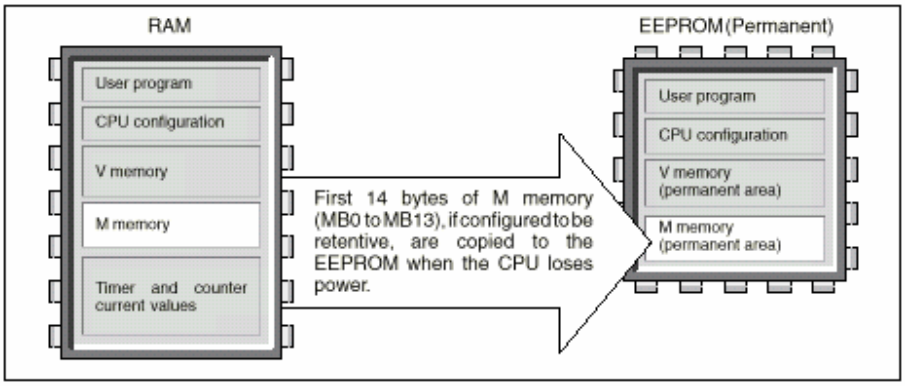
Sau đây chúng ta sẽ tìm hiểu chi tiết cơ chế hoạt động lưu giữ này.

Quá trình Download và Upload

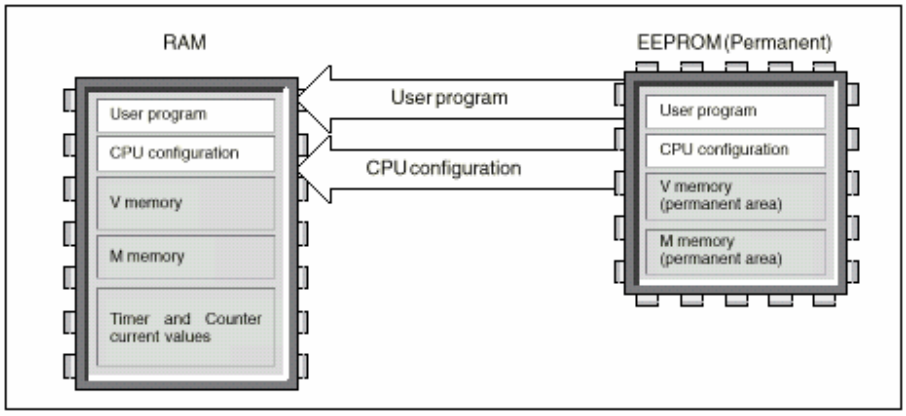




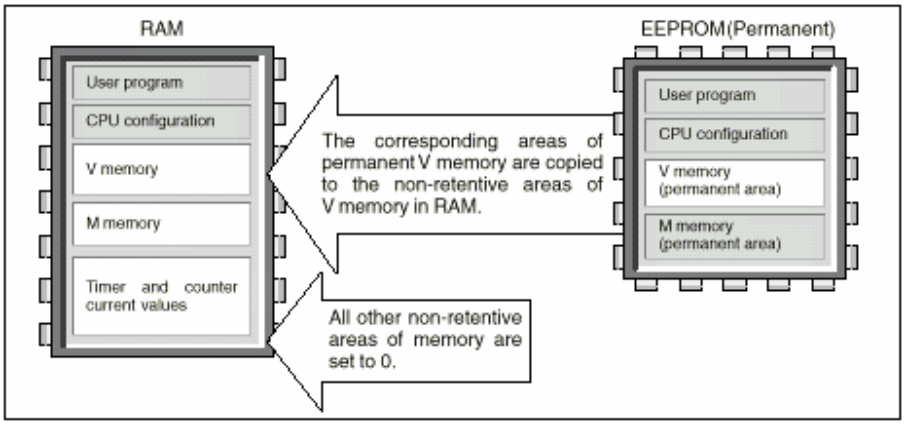
Khi CPU mất nguồn nuôi.



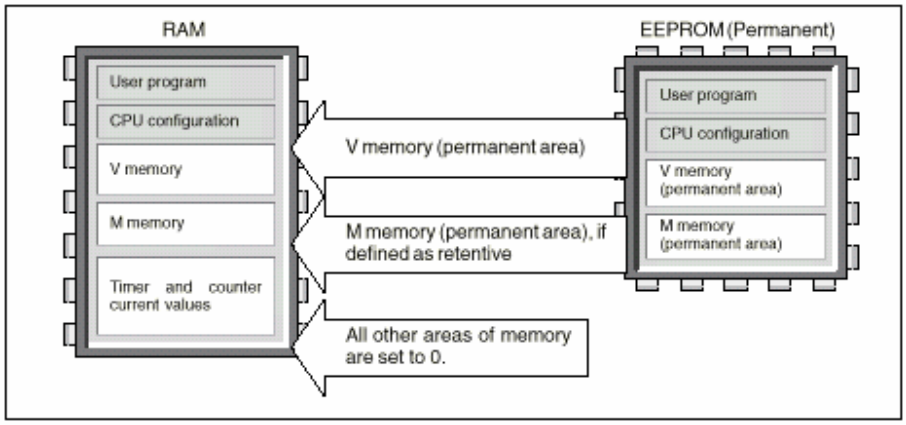
Khi CPU được cấp nguồn.



Nếu dữ liệu trong RAM vẫn còn được lưu giữ tốt:

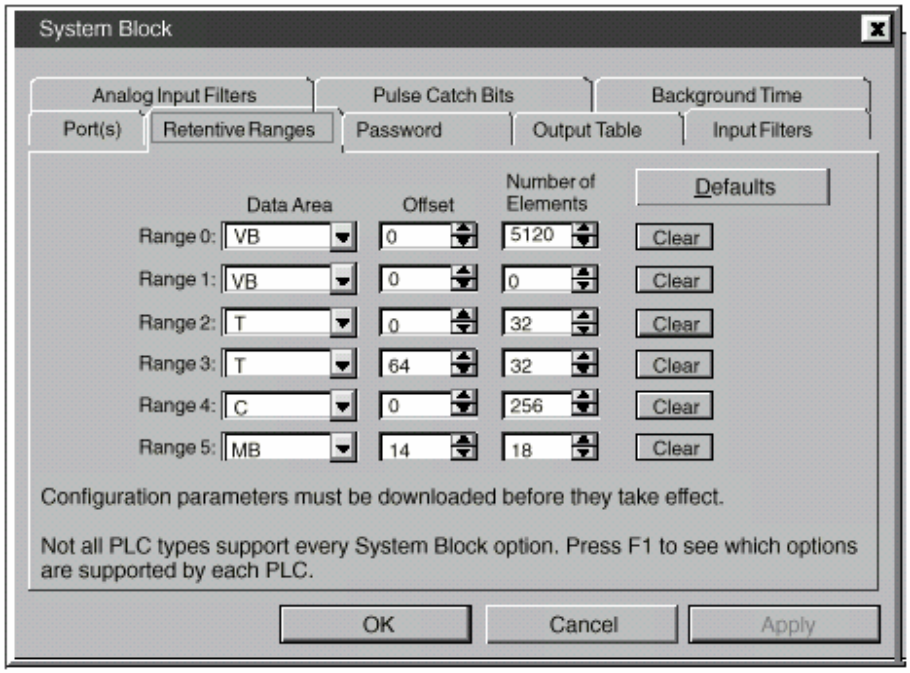


Nếu dữ liệu trong RAM không còn (thời gian mất nguồn nuôi quá lâu), CPU xóa toàn bộ bộ nhớ RAM, đặt bit SM0.2 = 1 trong vòng quét đầu tiên và khôi phục dữ liệu phần được lưu trữ:



Định nghĩa bộ nhớ dữ liệu cần lưu giữ.

Trên đây chúng ta nhận thấy rằng, bộ nhớ dữ liệu không phải toàn bộ đều được lưu giữ trong EEPROM mà chỉ một phần, được định nghĩa như là phần "retentive". Phần này được định nghĩa bằng cách chọn thực đơn **View > System Block** và chọn trang Retentive Ranges:



Chú ý: vùng M mặc định được xem là "non-retentive", khi đó không sử dụng đặc điểm lưu giữ dữ liệu lúc mất điện có nói đến trên đây. Phần được lưu giữ trong các vùng T và C (nếu được định nghĩa) là những giá trị đếm, các bit trạng thái không được lưu giữ. Trong vùng T chỉ được phép định nghĩa những timer dạng TONR, không phải TON (xem cụ thể ở phần lệnh, chương 8).

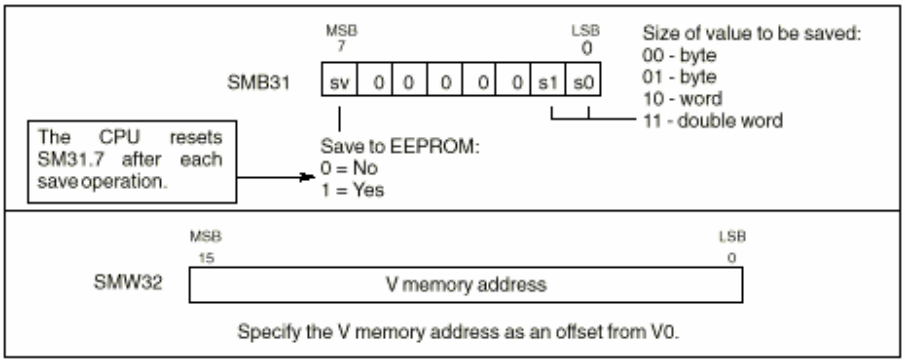
5.4 Lưu dữ liệu lâu dài từ chương trình

Trong khi chương trình đang thực thi, ta có thể lưu một giá trị (byte, từ đơn hoặc từ kép) trong vùng V vào EEPROM. Điều này cho phép cập nhật giá trị lưu giữ. Động tác này kéo dài thời gian vòng quét khoảng 5ms, chú ý nó không cập nhật giá trị trong cartridge.

Cách lưu giữ như sau:

- Đặt địa chỉ byte bắt đầu của giá trị cần lưu giữ (ví dụ 200 đối với VD200) vào SMW32.
- Đặt kích thước giá trị cần lưu giữ (byte, từ đơn hay từ kép) vào các bit SM31.0 và SM31.1 (xem hình minh họa phía sau).

- Đặt SM31.7 = 1.



Cuối mỗi vòng quét, CPU kiểm tra nếu SM31.7 = 1 thì thực hiện lưu và xóa bit SM31.7 = 0 sau khi lưu xong. Chú ý không được thay đổi giá trị cần lưu khi quá trình lưu chưa kết thúc (bit SM31.7 chưa bằng 0).

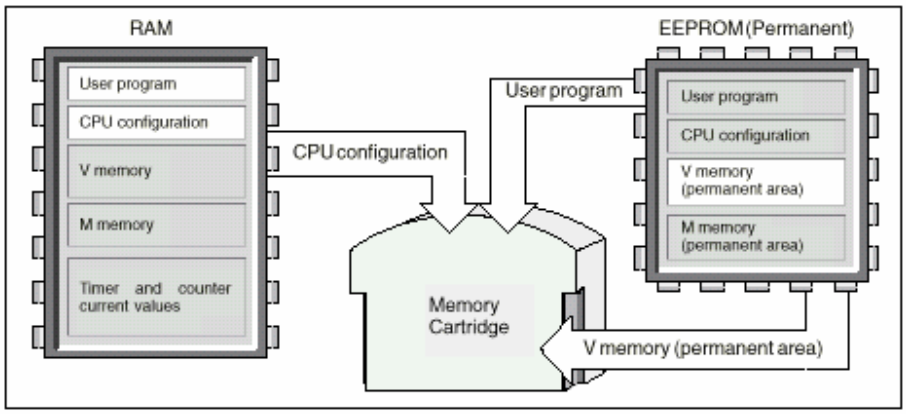
Không nên quá lạm dụng đặc tính này của S7-200 vì số lần lưu vào bộ nhớ kiểu EEPROM bị hạn chế (tuổi thọ EEPROM là khoảng 1 000 000 lần, ít nhất cũng là 100 000 lần). Như vậy nếu chúng ta lưu mỗi vòng quét một lần thì có thể tuổi thọ EEPROM chỉ kéo dài không tới 1 giờ rưỡi (tính cho thời gian vòng quét là khoảng 50 ms), tronh khi tuổi thọ đó có thể là 11 năm nếu chúng ta chỉ lưu mỗi giờ một lần.

5.5 Sử dụng Cartridge để lưu giữ chương trình

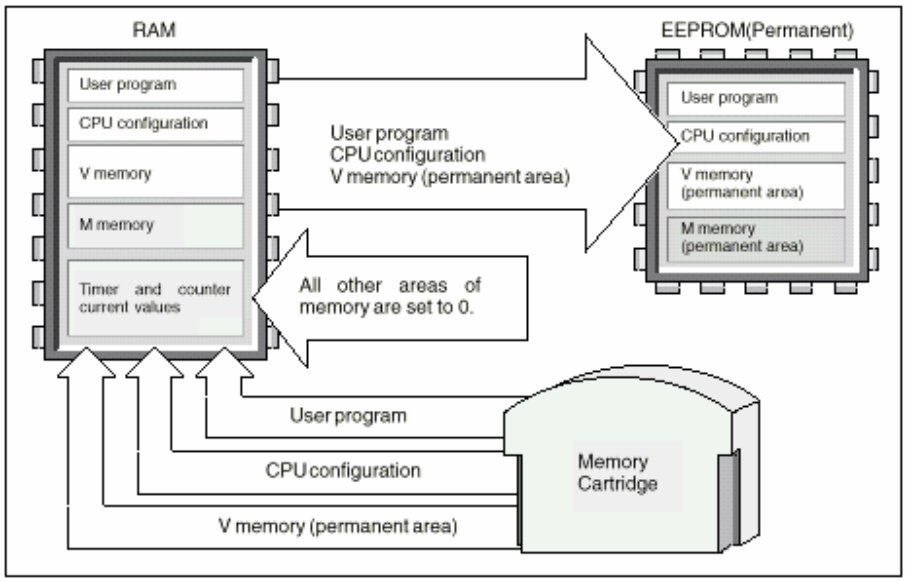
S7-200 cung cấp khả năng cắm thêm cartridge để lưu giữ chương trình, cấu hình CPU và dữ liệu trong vùng V của EEPROM (nghĩa là retentive V).

Ta có thể lưu chương trình từ RAM vào cartridge chỉ khi nào CPU được cấp nguồn và ở chế độ STOP. Chú ý trong cất giữ và thao tác với cartridge, nó có thể bị hỏng do phóng điện tĩnh điện.

Chúng ta có thể gắn cartridge trong khi CPU đang được cấp nguồn, nếu chương trình chưa có trong CPU thì phải nạp (download) vào CPU. Sau đó sử dụng thực đơn **PLC > Program Memory Cartridge** để lưu vào cartridge. Thông thường chúng ta cất giữ cartridge vào một nơi an toàn.



Chương trình được phục hồi từ cartridge bằng cách bật nguồn cho CPU với cartridge cắm sẵn. Trong trường hợp này bộ nhớ RAM bị xóa, nội dung cartridge được chép vào RAM, sau đó chương trình, cấu hình CPU và phần retentive V sẽ được chép vào EEPROM.



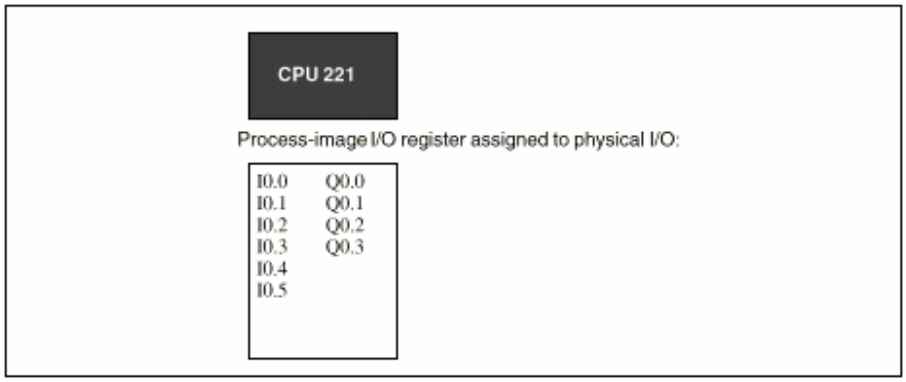
Chú ý: CPU được bật nguồn với một cartridge rỗng hay không đúng loại sẽ gây lỗi. Cartridge được lưu bởi CPU 221 và 222 có thể đọc được bởi CPU 224 nhưng không thể áp dụng ngược lại.

6. CÁC ĐẦU VÀO, RA

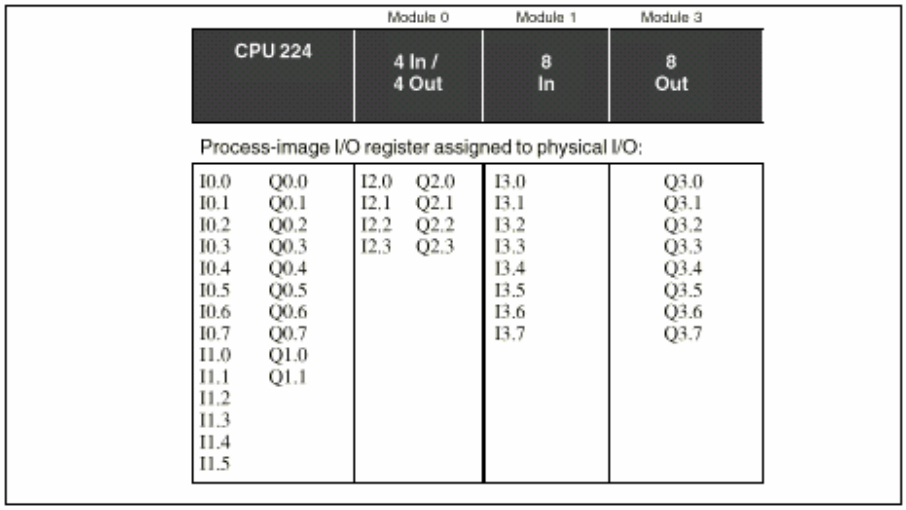
Các đầu vào ra chính là các điểm điều khiển của một hệ thống: các đầu vào phản ảnh trạng thái các thiết bị như các đầu dò, các công tắc, . . . và các đầu ra điều khiển những bộ phận chấp hành như mô tơ, bơm, van, . . .

6.1 Các đầu vào ra cục bộ và mở rộng

Cấu trúc MODULE của S7-200 tạo sự linh hoạt tối đa trong việc sử dụng để giải quyết các bài toán khác nhau. Nó cho phép chúng ta chọn số đầu vào ra tối ưu về mặt kinh tế. Một ưu điểm của S7-200 là những đầu vào ra cục bộ, nằm ngay trên CPU, thích hợp cho các bài toán đòi hỏi số đầu vào ra tối thiểu. Những đầu vào ra cục bộ có địa chỉ cố định. Ví dụ:



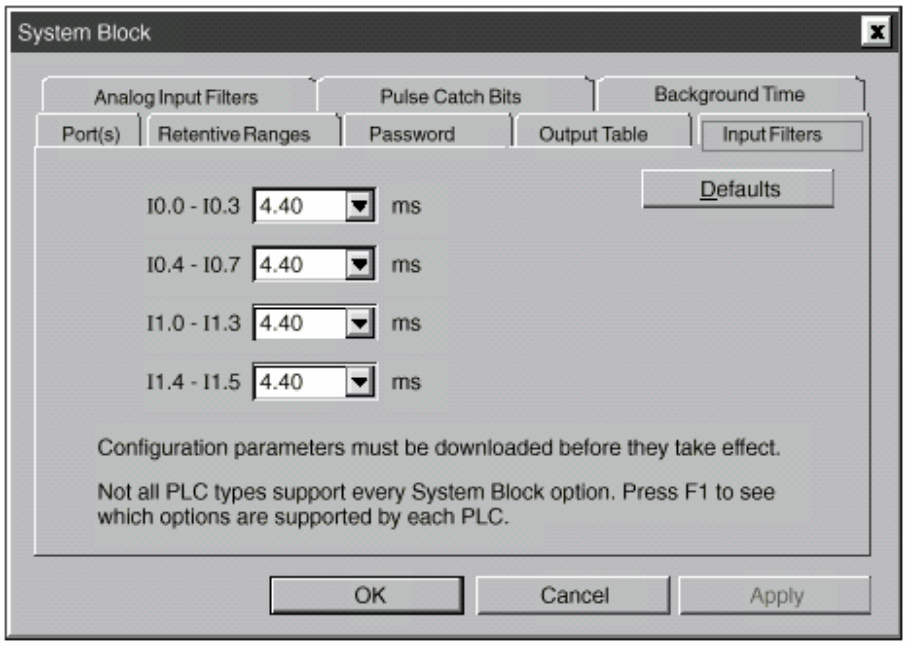
Chúng ta có thể tăng số đầu vào ra bằng các module mở rộng, ví dụ:



Các module mở rộng này được cắm nối tiếp nhau vào bên phải CPU. Địa chỉ các đầu vào ra trên các module mở rộng được tính liên tiếp, riêng cho từng loại (vào, ra, vào tương tự, ra tương tự) không ảnh hưởng lẫn nhau. Các đầu vào ra rời rạc được định địa chỉ chẵn byte, nghĩa là trên một module phải bắt đầu bằng x.0, x.1, . . . còn các đầu vào ra tương tự được định địa chỉ theo từ đơn, cách hai, nghĩa là bằng các số chẵn: AIW0, AIW2, AIW4, . . . AQW0, AQW2, AQW4, . . .

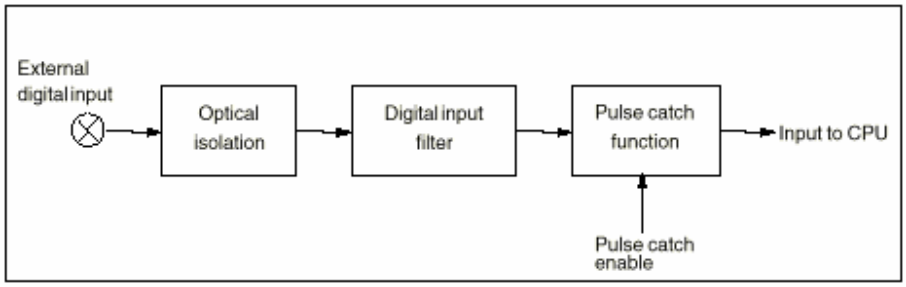
6.2 Lọc đầu vào

S7-200 có khả năng lọc các đầu vào rời rạc (chỉ các đầu cục bộ) bằng thời gian trễ để loại trừ hiện tượng nhiễu xung (có thể chọn từ 0.2 ms đến 12.8 ms). Tất nhiên, điều đó sẽ làm chậm tín hiệu vào. Chúng ta có thể đặt thời gian trễ thích hợp cho từng nhóm 04 đầu vào trong cấu hình của CPU bằng cách chọn thực đơn **View > System Block** và chọn trang Input Filters:

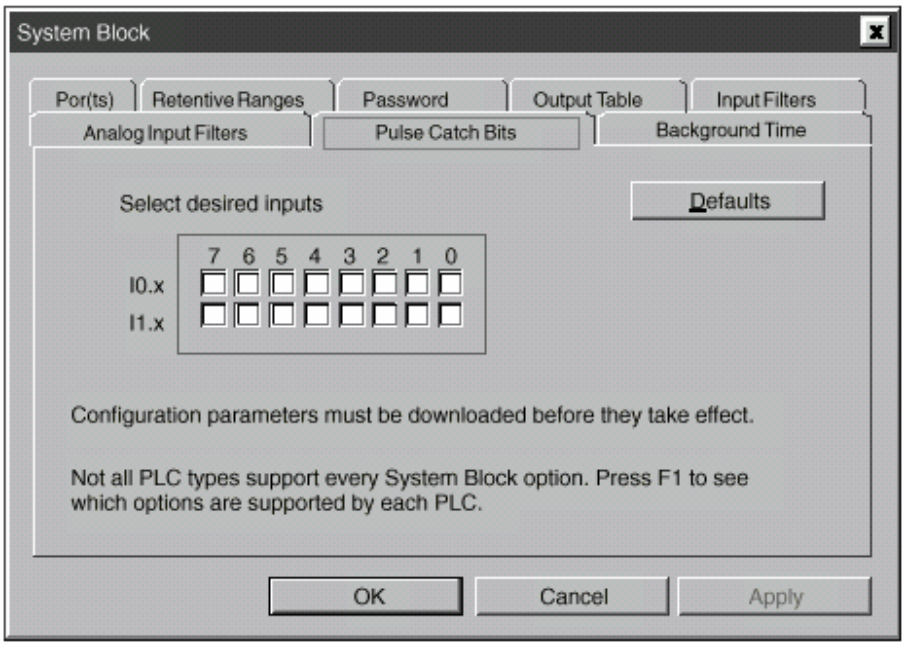


6.3 Nhận biết xung vào

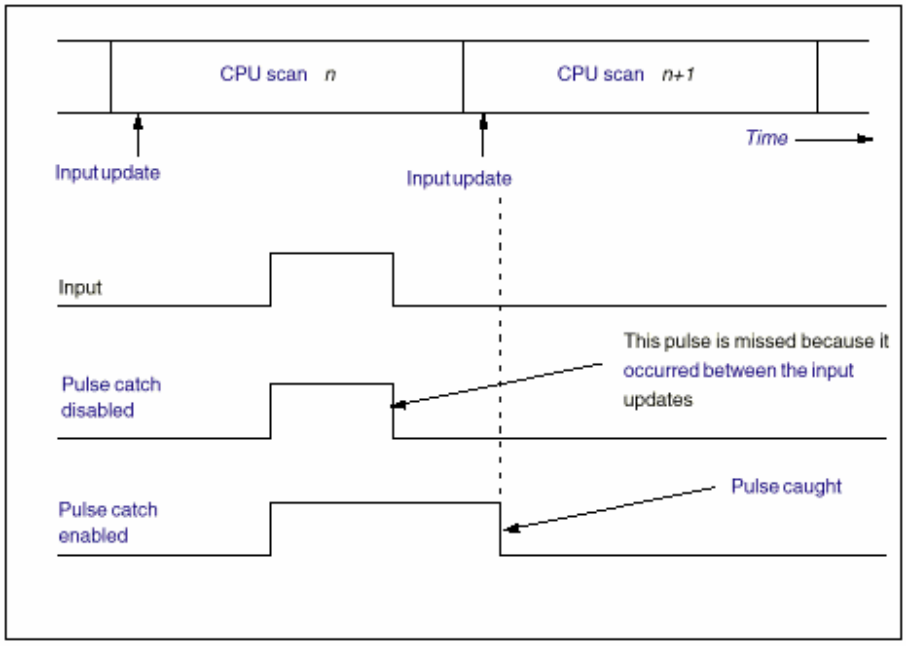
Bên cạnh vấn đề lọc đầu vào, chúng ta có thể nêu vấn đề một cách lô gic: PLC có thể bỏ qua những xung quá ngắn ở đầu vào ngoài ý muốn của chúng ta. Bởi vì chúng ta đã biết CPU chỉ cập nhật các đầu vào mỗi vòng quét một lần. S7-200 khắc phục điểm yếu này bằng chức năng "pulse catch":

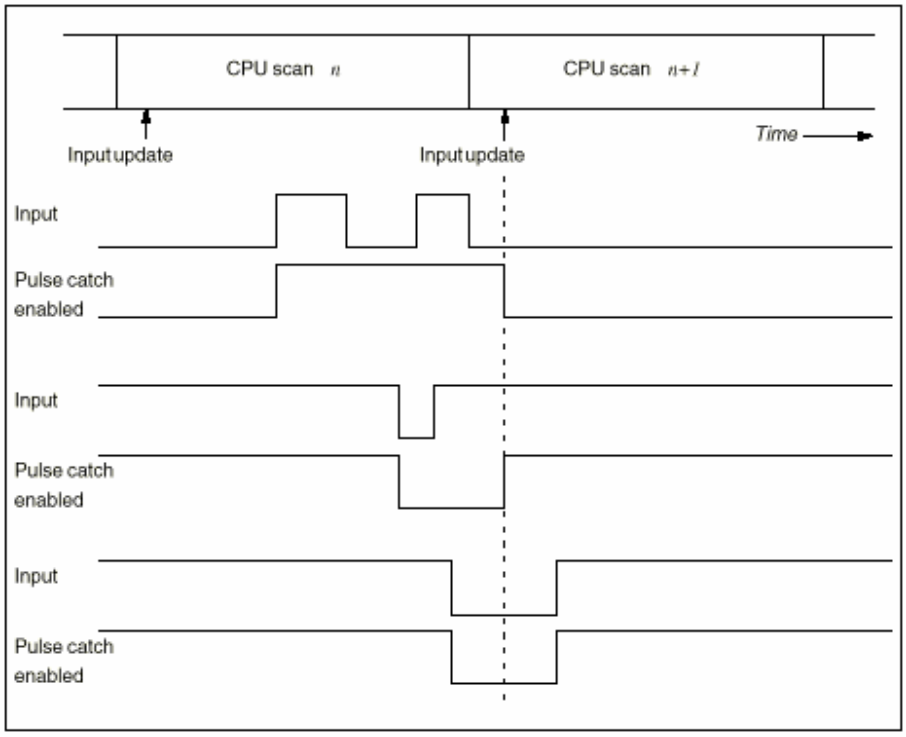


Ta có thể bật hoặc tắt chức năng này cho mỗi đầu vào cục bộ trong cấu hình CPU từ thực đơn **View > System Block**, chọn trang Pulse Catch Bits:



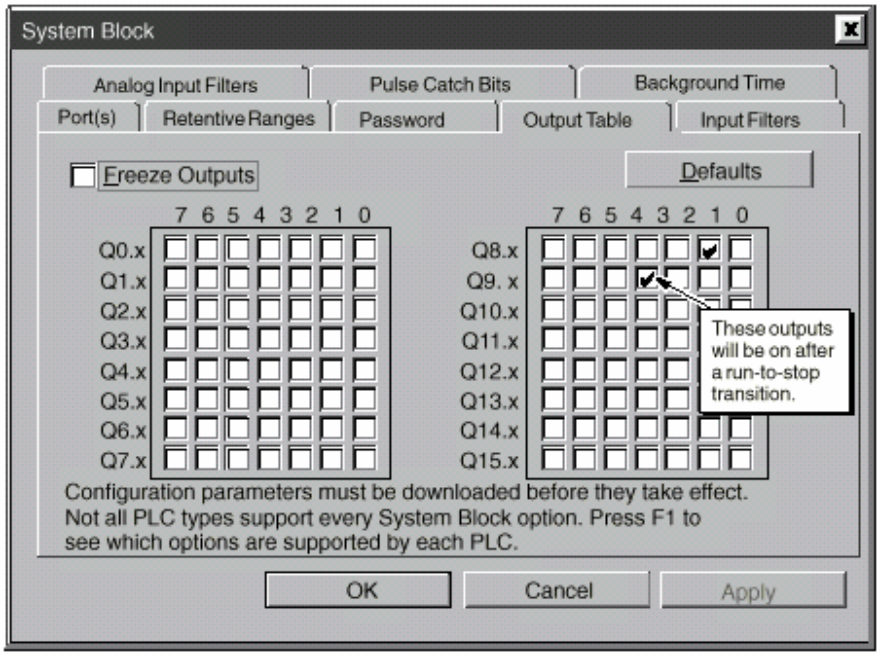
Sau đây là các ví dụ minh họa cho tính năng này:





6.4 Bảng các đầu ra

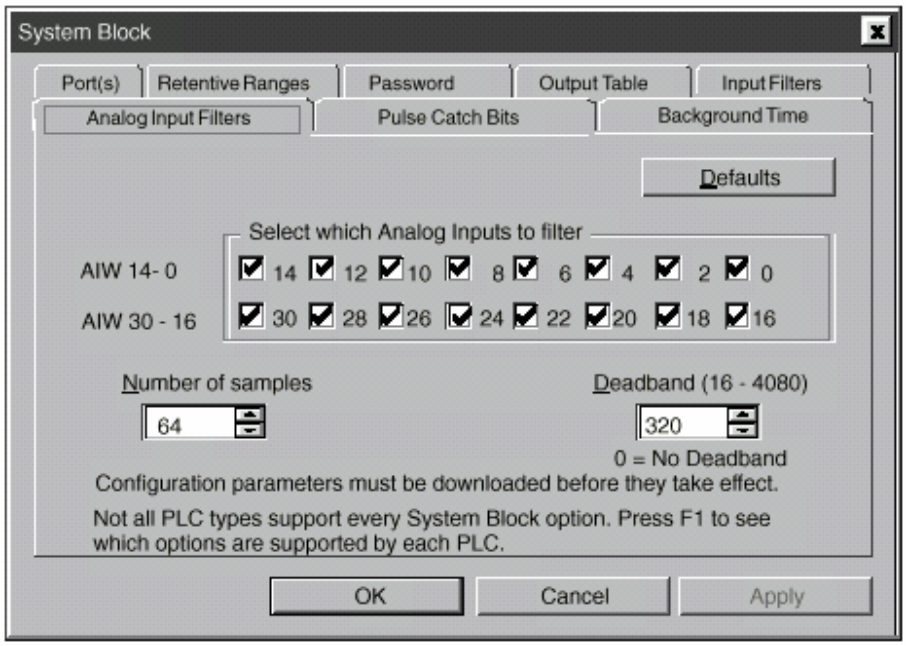
Bảng các đầu ra qui định trạng thái cho các đầu ra rời rạc khi CPU chuyển từ chế độ RUN sang chế độ STOP (bằng 0, 1 hay giữ nguyên trạng thái). Điều này rất quan trọng vì mục đích an toàn. Chúng ta định nghĩa bảng các đầu ra trong cấu hình của CPU bằng cách chọn thực đơn **View > System Block** và chọn trang Output Table:



6.5 Lọc đầu vào tương tự

Các đầu vào tương tự, cũng như các đầu vào rời rạc, có thể được lọc để chống hiện tượng nhiễu. Bản chất bộ lọc của một đầu vào tương tự là phép tính giá trị trung bình một số hữu hạn các giá trị lấy mẫu liên tiếp, nhằm giảm tác động của các giá trị ngoại lai. Tất nhiên tác động của bộ lọc bao giờ cũng làm chậm tín hiệu, trong trường hợp này có thể không thích hợp nếu đầu vào biển đổi nhanh. S7-200 xử lý vấn đề đó bằng khái niệm "deadband": nếu giá trị lấy mẫu vượt ra ngoài khoảng qui định so với giá trị trung bình thì bộ lọc không tính giá trị trung bình nữa mà cập nhật luôn giá trị mới.

Trong mọi trường hợp, người lập trình có thể bật hay tắt chức năng lọc cho từng đầu vào theo yêu cầu và cũng có thể đặt thông số chung cho các bộ lọc tương tự (số giá trị để tính trung bình, deadband) thông qua thực đơn **View > System Block** với trang Analog Input Filters:



6.6 Vào ra tốc độ cao

Khác với các vi mạch điện tử, các mạch điều khiển tự động thông thường hoạt động với tốc độ thấp hơn. Tuy nhiên, thỉnh thoảng chúng ta cũng cần ghi nhận và xử lý những biến đổi tốc độ cao. S7-200 đáp ứng yêu cầu này bằng các đầu vào và các bộ đếm tốc độ cao cũng như bằng đầu ra xung tốc độ cao.

Các bộ đếm tốc độ cao

Các bộ đếm tốc độ cao trong S7-200 có khả năng đếm những tần số đến 20 kHz với nhiều chế độ hoạt động khác nhau:

- HSC0 và HSC4 có thể hoạt động ở một trong 08 chế độ, có thể đếm các đầu vào một pha hoặc hai pha.
- HSC1 và HSC2 có 12 chế độ hoạt động, với các đầu vào một pha cũng như hai pha.
- HSC3 và HSC5 là những bộ đếm đơn giản, với một chế độ hoạt động và chỉ đếm đầu vào một pha.

Chi tiết hơn về các bộ đếm tốc độ cao sẽ được nói đến ở phần lệnh cụ thể (chương 8). Hai bảng sau tóm tắt về các bộ đếm này:

Chúng ta nhận thấy rằng nếu sử dụng HSC0 trong những chế độ từ 3 đến 10 thì không thể sử dụng HSC3 bởi vì HSC0 và HSC3 cả hai đều dùng đầu vào I0.1. Tương tự như thế đối với HSC4 và HSC5.

Các đầu vào từ I0.0 đến I0.3 còn có thể được sử dụng làm các đầu vào gây ngắt, do đó chúng ta cần chú ý không sử dụng chúng vừa làm các đầu vào gây ngắt, vừa làm các đầu vào cho các bộ đếm tốc độ cao cung một lúc.

Mode	HSC0			HSC3	HSC4			HSC5
	I0.0	I0.1	I0.2	I0.1	I0.3	I0.4	I0.5	I0.4
0	Clk	-	-	Clk	Clk	-	-	Clk
1	Clk	-	Reset	-	Clk	-	Reset	-
2	-	-	-	-	-	-	-	-
3	Clk	Direction	-	-	Clk	Direction	-	-
4	Clk	Direction	Reset	-	Clk	Direction	Reset	-
5	-	-	-	-	-	-	-	-
6	Clk Up	Clk Down	-	-	Clk Up	Clk Down	-	-
7	Clk Up	Clk Down	Reset	-	Clk Up	Clk Down	Reset	-
8	-	-	-	-	-	-	-	-
9	Phase A	Phase B	-	-	Phase A	Phase B	-	-
10	Phase A	Phase B	Reset	-	Phase A	Phase B	Reset	-
11	-	-	-	-	-	-	-	-

Nói chung, một đầu vào không thể được sử dụng cho hai mục đích cùng một lúc. Tuy nhiên, nếu không được sử dụng trong mục đích này, chúng có thể được tận dụng cho mục đích kia. Ví dụ nếu HSC0 đang hoạt động ở chế độ 2, chỉ sử dụng I0.0 và I0.2 thì I0.1 vẫn có thể được khai thác bởi ngắt hay HSC3.

Mode	HSC1				HSC2			
	I0.6	I0.7	I1.0	I1.1	I1.2	I1.3	I1.4	I1.5
0	Clk	-	-	-	Clk	-	-	-
1	Clk	-	Reset	-	Clk	-	Reset	-
2	Clk	-	Reset	Start	Clk	-	Reset	Start
3	Clk	Direction	-	-	Clk	Direction	-	-
4	Clk	Direction	Reset	-	Clk	Direction	Reset	-
5	Clk	Direction	Reset	Start	Clk	Direction	Reset	Start
6	Clk Up	Clk Down	-	-	Clk Up	Clk Down	-	-
7	Clk Up	Clk Down	Reset	-	Clk Up	Clk Down	Reset	-
8	Clk Up	Clk Down	Reset	Start	Clk Up	Clk Down	Reset	Start
9	Phase A	Phase B	-	-	Phase A	Phase B	-	-
10	Phase A	Phase B	Reset	-	Phase A	Phase B	Reset	-
11	Phase A	Phase B	Reset	Start	Phase A	Phase B	Reset	Start

Hai bộ đếm HSC1 và HSC2 hoạt động hoàn toàn độc lập với nhau, có thể khai thác tối đa cả hai cùng một lúc mà không hề ảnh hưởng lẫn nhau.

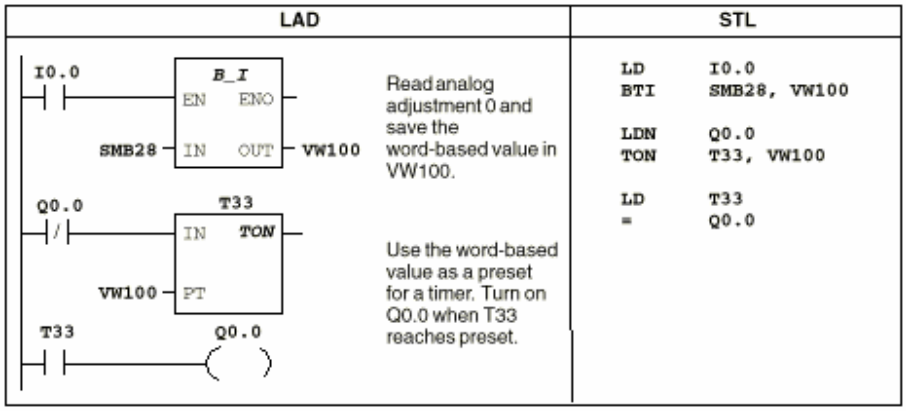
Đầu ra xung tốc độ cao

S7-200 cho phép sử dụng Q0.0 và Q0.1 như những đầu ra phát xung tốc độ cao, dạng PTO hoặc PWM. Chi tiết xem chương 8, sau đây là vài nét sơ lược:

- Xung kiểu PTO (Pulse Train Output) là sóng vuông, 50% chu kỳ có giá trị 0, 50% chu kỳ có giá trị 1. Có thể định nghĩa số xung phát ra nằm trong khoảng từ 1 xung đến 4 294 967 295 xung. Chu kỳ có thể xác định theo độ phân giải là micro giây hoặc mili giây với giá trị từ 50 μs đến 65 535 μs hay từ 2 ms đến 65 535 ms. Lưu ý nên chọn chu kỳ là số chẵn, chu kỳ là số lẻ có thể gây biến dạng sóng. S7-200 còn cho phép tạo dãy xung PTO với chu kỳ biến thiên theo qui luật nào đó, rất thích hợp trong điều khiển động cơ bước.
- Xung kiểu PWM (Pulse Width Modulation) có chu kỳ cố định và độ rộng xung (thời gian có giá trị bằng 1) thay đổi. Cả hai giá trị này đều có thể xác định theo độ phân giải là micro giây hoặc mili giây. Chu kỳ xung có thể nằm trong khoảng từ 50 μs đến 65 535 μs hay từ 2 ms đến 65 535 ms. Độ rộng xung có thể nằm trong khoảng từ 0 μs đến 65 535 μs hay từ 0 ms đến 65 535 ms. Nếu độ rộng xung bằng chu kỳ, đầu ra luôn luôn bằng 1. Nếu độ rộng xung bằng 0, đầu ra luôn luôn bằng 0.

6.7 Đinh chỉnh tương tự

S7-200 CPU có 1 hoặc 2 đinh chỉnh tương tự phía trước. Ta có thể vặn chúng theo chiều kim đồng hồ hay ngược lại trong khoảng 270° để tăng hay giảm giá trị tương ứng với chúng là các byte trong SMB28 và SMB29. Như vậy những giá trị này có thể thay đổi trong khoảng từ 0 đến 255 và chương trình có thể sử dụng chúng như những giá trị chỉ đọc, thay đổi được theo sự can thiệp từ ngoài chương trình. Ví dụ:



7. CÁC PHƯƠNG THỨC TRUYỀN THÔNG

Truyền thông là phần khá phức tạp trong việc làm chủ PLC. Điều quan trọng là chúng ta phải nắm rõ các kiểu cấu trúc mạng khác nhau của các PLC, các phương thức truyền thông được sử dụng và làm chủ tất cả các thành phần cấu thành nên mạng. Chúng ta không đi sâu vào chi tiết trong tài liệu này mà chỉ điểm qua những nét chính.

Trước hết, ta phân biệt một số **mô hình mạng**:

- Mạng đơn chủ (Single Master)
- Mạng đa chủ (Multiple Master)
- Sử dụng Modem 10 bit nối 01 chủ với 01 PLC S7-200 hoạt động như trạm (Slave)
- Sử dụng Modem 11 bit trong mạng đơn chủ

Ví dụ về cấu hình mạng:

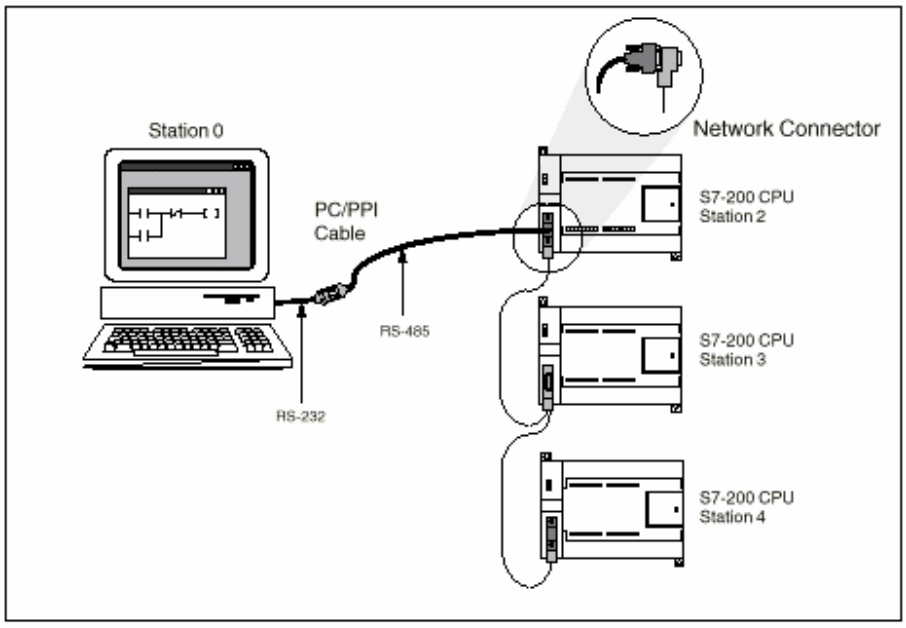


Figure 7-1 Using a PC/PPI Cable for Communicating with Several S7-200 CPUs

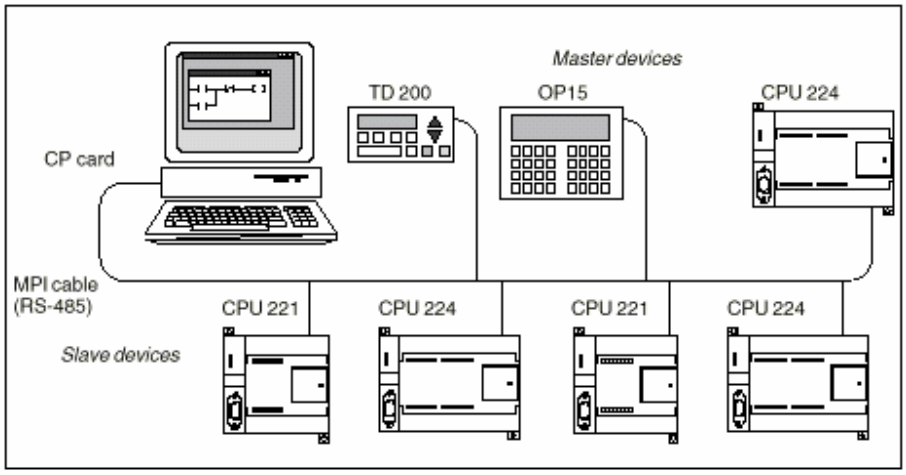


Figure 7-2 Example of a CP Card with Master and Slave Devices

Trong những thành phần tham gia mạng, các CPU có thể hoạt động như chủ hoặc như trạm; TD 200 là thiết bị chủ; thiết bị lập trình hoặc máy vi tính cài STEP 7 cũng là thiết bị chủ. Phần mềm STEP 7 - Micro / Win 32 được thiết kế chỉ kết nối được với một CPU S7-200 tại một thời điểm, tuy nhiên nó có thể kết nối tới bất cứ CPU nào trong mạng.

Các **phương thức truyền thông** chính:

- Điểm đối điểm: Point-to-Point Interface (PPI)
- Đa điểm: Multipoint Interface (MPI)

- PROFIBUS (Process Field Bus)

Các phương thức này đều đặt cơ sở trên cấu trúc OSI (Open System Interconnection) 7 lớp. Các phương thức PPI và MPI cũng sử dụng nguyên lý mạch hỏi vòng (Token ring), phù hợp với chuẩn PROFIBUS đã được qui định trong bộ chuẩn châu Âu EN 50170.

Những phương thức trên đều là bất đồng bộ, đơn vị cơ sở là ký tự với 01 start bit, 08 data bit, even parity và 01 stop bit. Khung dữ liệu bao gồm những ký tự đặc biệt mở đầu và kết thúc, địa chỉ nguồn (nơi gửi) và đích (nơi đến), độ dài dữ liệu và "checksum". Cả ba phương thức có thể cùng hoạt động chung trên một mạng, chỉ cần điều kiện cùng tốc độ truyền (baud rate).

Mạng theo chuẩn PROFIBUS sử dụng đường truyền là những cặp dây xoắn theo chuẩn RS-485. Chuẩn đường truyền này cho phép nối tới 32 thiết bị trên một bộ phận (segment). Khoảng cách giữa hai điểm xa nhất trong một bộ phận như vậy, tùy theo tốc độ đường truyền sử dụng, có thể lên đến 1200 m. Các bộ phận lại có thể nối với nhau qua những "repeater" để tăng số thiết bị trong mạng cũng như khoảng cách hoạt động cho đến 9600 m tùy theo tốc độ truyền.

Các phương thức này phân biệt 02 loại thiết bị: chủ và tớ (trạm). Thiết bị chủ có thể gửi yêu cầu lên mạng trong khi trạm chỉ trả lời, không bao giờ tự gửi thông tin lên mạng.

Số địa chỉ tối đa là 127 (0 đến 126) với nhiều nhất là 32 thiết bị chủ. Mỗi thiết bị trên mạng phải có địa chỉ khác nhau. Mặc định, thiết bị lập trình (hay PC) được định địa chỉ 0, các thiết bị giao diện như TD 200, OP3, OP7, ... có địa chỉ là 1 còn PLC được định địa chỉ mặc định là 2.

Phương thức PPI

PPI là phương thức chủ / tớ. Các thiết bị chủ (CPU, thiết bị lập trình, TD 200) gửi yêu cầu đến các trạm và các trạm trả lời. Các trạm không bao giờ tự gửi thông tin lên mạng mà chỉ chờ nhận các yêu cầu của các thiết bị chủ để trả lời. Tất cả các CPU S7-200 đều có thể hoạt động như trạm trong mạng.

Một số CPU có thể hoạt động như thiết bị chủ trong mạng khi ở chế độ RUN, nếu chương trình bật chế độ PPI master (với SMB30). Một khi ở trong chế độ này, ta có thể đọc hay viết vào một CPU khác bằng các lệnh NETR và NETW. Trong khi đó CPU vẫn trả lời các thiết bị chủ khác như một trạm thông thường. PPI không hạn chế số thiết bị chủ được phép nối với một trạm, tuy nhiên như trên đã nêu, số thiết bị chủ tối đa trong một mạng là 32.

Phương thức MPI

MPI có thể là phương thức chủ / tớ hay chủ / chủ. Cách thức hoạt động phụ thuộc vào loại thiết bị. Chẳng hạn nếu thiết bị đích là CPU S7-300 thì MPI tự động trở thành

chủ / chủ bởi vì các CPU S7-300 là các thiết bị chủ trong mạng. Nhưng nếu đích là CPU S7-200 thì MPI lại là chủ / tớ vì các CPU S7-200 lúc đó được coi như là trạm.

Khi hai thiết bị trong mạng kết nối với nhau bằng phương thức MPI, chúng tạo nên một liên kết riêng, không thiết bị chủ nào khác có thể can thiệp vào liên kết này. Thiết bị chủ trong hai thiết bị kết nối thường giữ mối liên kết đó trong một khoảng thời gian ngắn hoặc hủy liên kết vô thời hạn (giải phóng đường truyền).

Những liên kết như trên đòi hỏi một tài nguyên nhất định trong CPU nên mỗi CPU chỉ có thể hỗ trợ một số lượng hữu hạn các liên kết như vậy. Thông thường một CPU cho phép 04 liên kết, 02 trong đó một dành riêng cho thiết bị lập trình hay PC, một dành cho giao diện. Điều này cho phép lúc nào cũng có thể kết nối ít nhất một thiết bị lập trình hoặc PC, một giao diện với CPU. Những thiết bị chủ khác (như các CPU khác chẳng hạn) không thể kết nối qua các liên kết dành riêng này.

Các CPU S7-300 và S7-400 có thể kết nối với các CPU S7-200 bằng một trong hai liên kết còn lại của CPU S7-200 và đọc hay ghi dữ liệu vào CPU S7-200 với các lệnh XGET và XPUT.

Phương thức PROFIBUS

Phương thức PROFIBUS được thiết kế cho việc truyền thông tốc độ cao với các thiết bị phân phối vào ra, thường cũng được gọi là các đầu vào ra từ xa (remote I/O). Những thiết bị như vậy được nhiều nhà sản xuất cung cấp, từ các module vào ra đơn giản đến các bộ điều khiển mô tơ và các PLC.

Mạng PROFIBUS thường bao gồm một thiết bị chủ và nhiều trạm vào ra. Thiết bị chủ được đặt cấu hình để nhận biết loại cũng như địa chỉ của các trạm nối vào nó. Sau đó nó sẽ tự kiểm tra các trạm theo cấu hình được đặt. Thiết bị chủ ghi vào các trạm và đọc dữ liệu từ đó một cách liên tục. Nói chung mỗi thiết bị chủ thường làm chủ các trạm của mình, các thiết bị chủ khác trên mạng (nếu có) chỉ có thể truy cập rất hạn chế vào các trạm không phải của chúng.

Phương thức định nghĩa bởi người sử dụng (FreePort)

Phương thức này cho phép người lập trình làm chủ việc truyền thông, thực tế là định nghĩa phương thức truyền thông riêng, có thể kết nối tới nhiều loại thiết bị thông minh khác.

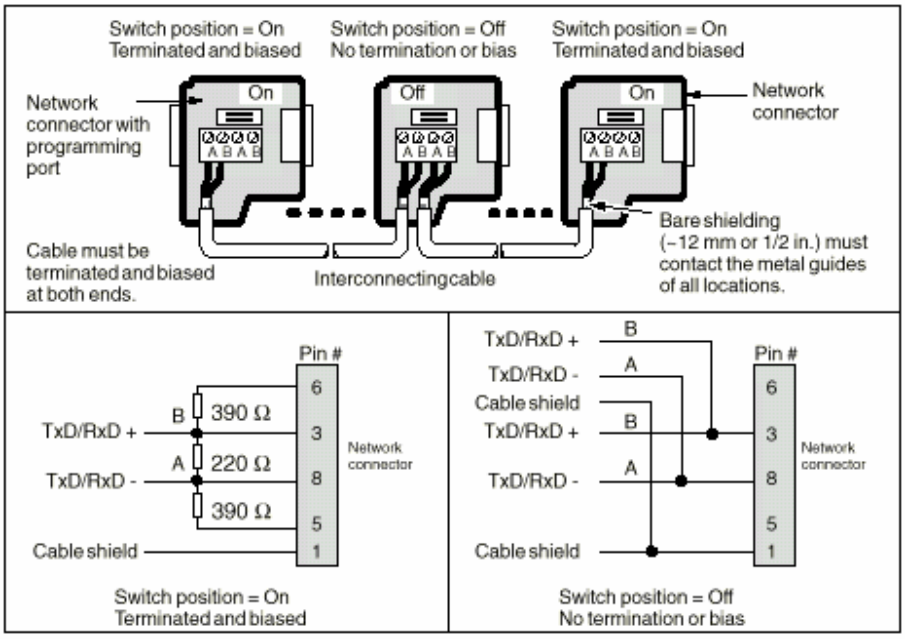
Chương trình kiểm soát cổng truyền thông trong phương thức này thông qua các ngắt nhận, ngắt gửi, lệnh nhận (RCV) và lệnh gửi (XMT). Cách thức truyền thông hoàn toàn do chương trình làm chủ. Phương thức này được điều khiển với byte SMB30 (dành cho cổng 0) và chỉ hoạt động trong chế độ RUN. Khi CPU chuyển sang chế độ STOP, phương thức này bị hủy và cổng truyền thông trở về phương thức bình thường PPI.

Cuối cùng chúng ta xét đến **cấu hình phần cứng** của mạng:

Do phần này nặng về tính kỹ thuật và đòi hỏi tính chính xác trong từng trường hợp cụ thể nên chúng ta sẽ không nói đến kỹ trong tài liệu này. Sơ lược như ta đã biết, đường dây truyền tuân theo chuẩn RS 485, bản chất là cặp dây xoắn:

General Features	**Specification**
Type	Shielded, twisted pair
Conductor cross section	24 AWG (0.22 mm 2) or larger
Cable capacitance	< 60 pF/m
Nominal impedance	100 Ω to 120 Ω

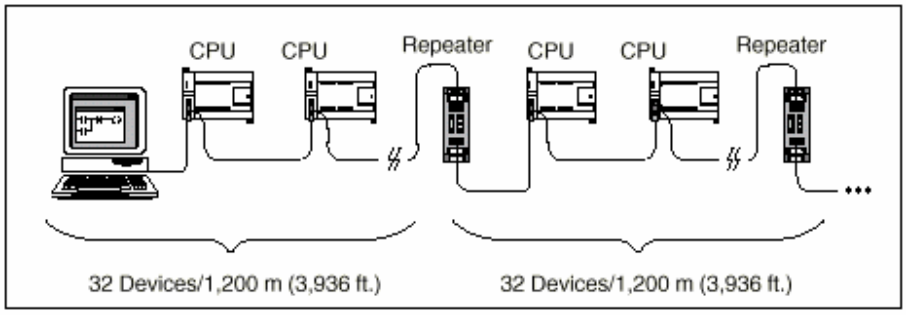
cách đấu nối như những mạng sử dụng Token ring (mạch hỏi vòng) thông thường:



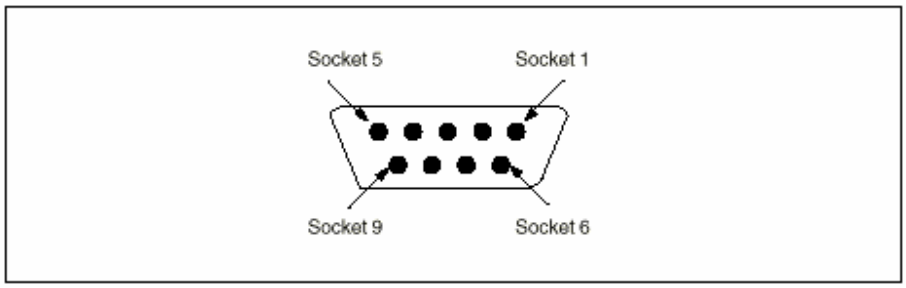
khoảng cách truyền tối đa giới hạn tùy theo tốc độ truyền:

Transmission Rate	**Maximum Cable Length of a Segment**
9.6 kbaud to 19.2 kbaud	1,200 m (3,936 ft.)
187.5 kbaud	1,000 m (3,280 ft.)

có thể dùng bộ lặp để tăng khoảng cách cũng như số thiết bị:



Cổng truyền thông trên CPU S7-200:

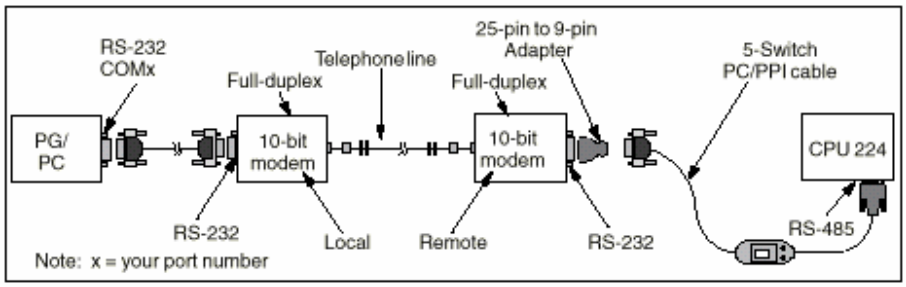


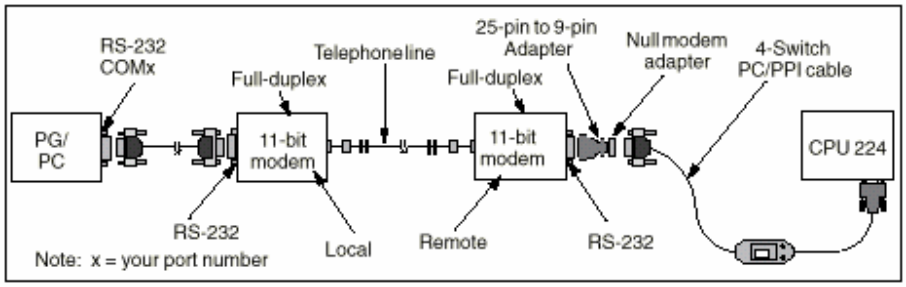
Socket	PROFIBUS Designation	Port 0
1	Shield	Logic common
2	24 V Return	Logic common
3	RS-485 Signal B	RS-485 Signal B
4	Request-to-Send	RTS (TTL)
5	5 V Return	Logic common
6	+5 V	+5 V, 100 Ω series resistor
7	+24 V	+24 V
8	RS-485 Signal A	RS-485 Signal A
9	Not applicable	10-bit protocol select (input)
Connector shell	Shield	Chassis ground

Vấn đề cuối cùng là kết nối PC với mạng RS 485:

Hardware Supported	Type	Baud Rate Supported	Comments
PC/PPI cable	Cable connector to PC comm port	9.6 kbaud 19.2 kbaud	Supports PPI protocol
CP 5511	Type II, PCMCIA-card	9.6 kbaud 19.2 kbaud 187.5 kbaud	Supports PPI, MPI, and PROFIBUS protocols for notebook PCs
CP 5611	PCI-card (version 3 or greater)		Supports PPI, MPI, and PROFIBUS protocols for PCs
MPI	Integrated in PG PC ISA-card		

Ở đây chúng ta không đi sâu vào cách thiết lập thông số cho cáp PC/PPI cũng như các card CP hay MPI hoạt động. Chúng ta chỉ nói thêm một chút về cáp PC/PPI vì nó được sử dụng khá thông dụng mà chúng ta đã nhắc đến trong phần đầu của tài liệu này (chương 3). Đây là cáp chuyển đổi giữa hai chuẩn RS 485 và RS 232. Nếu nối với máy vi tính, đầu RS 232 được cắm vào cổng COM, chú ý với loại cáp có DIP switch 05 vị trí thì phải chọn DCE (Data Control Equipment). Cáp này còn được sử dụng để nối với Modem, cũng có giao tiếp RS 232 nhưng là DTE (Data Terminal Equipment) như các minh họa sau:





8. TẬP LỆNH SIMATIC

9. TẬP LỆNH IEC 1131-3

10. PHỤ LỤC