Chương 1

Giới thiệu về Micro PLC "CP1L/1H"

Phần I: Các khái niệm cơ bản

1.1 Các hệ đếm (Number System):

Bộ xử lý trung tâm (CPU) bên trong PLC chỉ làm việc với 2 trạng thái 0 hoặc 1 (dữ liệu số) hay ON/OFF, do đó cần thiết phải có một số cách biểu diễn các đại lượng liên tục thường gặp hàng ngày dưới dạng các dãy số 0 và 1.

- **Hệ nhị phân (Binary)**
- **Hệ thập phân (Decimal)**
- **Hệ thập lục (hay hệ hexa) (Hexadecimal)**

1. Hệ nhị phân (Binary)

Là hệ đếm trong đó chỉ sử dụng 2 con số là 0 và 1 để biểu diễn tất cả các con số và đại lượng. Dãy số nhị phân được đánh số như sau : bit ngoài cùng bên phải là bit 0, bit thứ hai ngoài cùng bên phải là bit 1, cứ như vậy cho đến bit ngoài cùng bên trái là bit n. Bit nhị phân thứ n có *trọng số* là 2^n x 0 hoặc 1, trong đó n = số của bit trong dãy số nhị phân, 0 hoặc 1 là giá trị của bit n đó. Giá trị của dãy số nhị phân bằng tổng trọng số của từng bit trong dãy. .

Ví dụ : Dãy số nhị phân 1001 sẽ có giá trị như sau :

$1001 = 1x2^3 + 0x\ 2^2 + 0x2^1 + 1x2^0 = 9$

2. Hệ thập phân (Decimal)

Là hệ đếm sử dụng 10 chữ số là 0 1 2 3 4 5 6 7 8 9 để biểu diễn các con số. Hệ thập phân còn kết hợp với hệ nhị phân để có cách biểu diễn gọi là BCD (Binary-Coded Decimal)

3. Hệ thập lục (Hexadecimal)

Là hệ đếm sử dụng 16 ký số là 0 1 2 3 4 5 6 7 8 9 A B C D E F (trong đó có 9 chữ số từ 0-10, các chữ số từ 11 đến 15 được biểu diễn bằng các ký tự từ A-F)

Khi viết, để phân biệt người ta thường thêm các chữ BIN (hoặc số $_2$), BCD hay HEX (hoặc h) vào các con số :

HEX	BCD	Số nhị phân 4 bit tương đương			
		Bit 3	Bit 2	Bit 1	Bit 0
		$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
0	0	0	0	0	0
1	1	0	0	0	1
2	2	0	0	1	0
3	3	0	0	1	1
4	4	0	1	0	0
5	5	0	1	0	1
6	6	0	1	1	0
7	7	0	1	1	1
8	8	1	0	0	0
9	9	1	0	0	1
A	-	1	0	1	0
B	-	1	0	1	1

C	**-**	**1**	**1**	**0**	**0**
D	**-**	**1**	**1**	**0**	**1**
E	**-**	**1**	**1**	**1**	**0**
F	**-**	**1**	**1**	**1**	**1**

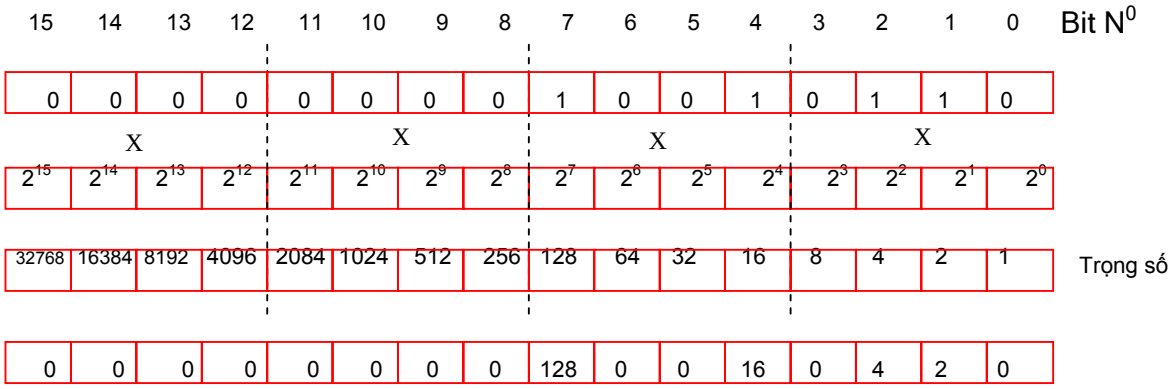
Bảng trên là cách biểu diễn của các chữ số hexa và BCD bằng các chữ số nhị phân (mỗi chữ số hexa và BCD đều cần 4 bit nhị phân).

BIN (Binary) = Nhị phân
BCD (Binary Coded Decimal)= Nhị thập phân
HEX (Hexadecimal) = Hệ thập lục (Hexa)

1.2 Cách biểu diễn số nhị phân

1.2.1) Biểu diễn số thập phân bằng số nhị phân

Ví dụ Giả sử ta có 16 bit như sau : 0000 0000 1001 0110
Để tính giá trị thập phân của 16 bit này ta làm như sau :



Như vậy : $0000\ 0000\ 1001\ 0110_2$ = 128 + 16 + 4 + 2
= # 150 (thập phân)

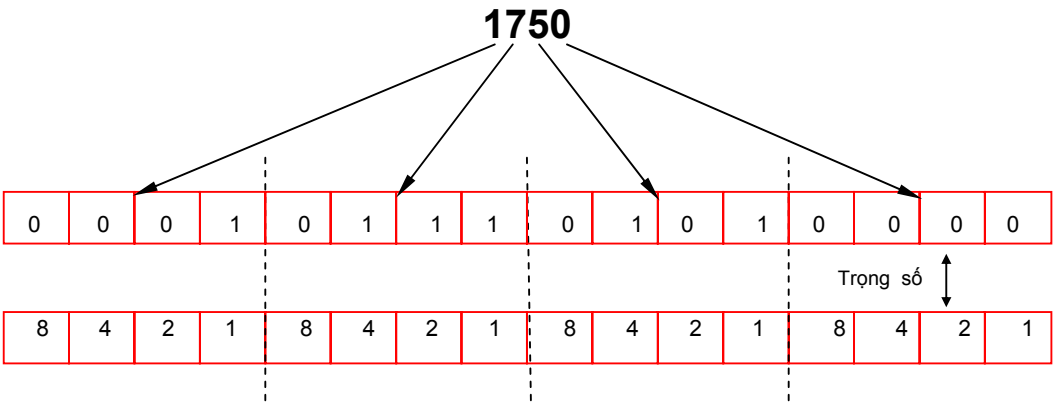
Ngược lại : $(1750)_{10}$ = (1024 + 512 + 128 + 64 + 16 + 4 + 2)
= $(0000\ 0110\ 1101\ 0110)_2$

Như trên ta thấy, việc tính nhẩm giá trị thập phân của một dãy số nhị phân dài là rất mất thời gian. Vì vậy người ta đã có một cách biểu diễn số thập phân dưới dạng đơn giản hơn. Đó là dạng BCD và được dùng phổ biến trong các loại PLC của OMRON.

1.2.2) Biểu diễn số nhị phân dưới dạng BCD

Khi biểu diễn bằng mã BCD, mỗi số thập phân được biểu diễn riêng biệt bằng nhóm 4 bit nhị phân.

Ví dụ: Giả sử ta có một số hệ thập phân là 1.750 và cần chuyển nó sang dạng mã BCD 16 bit.

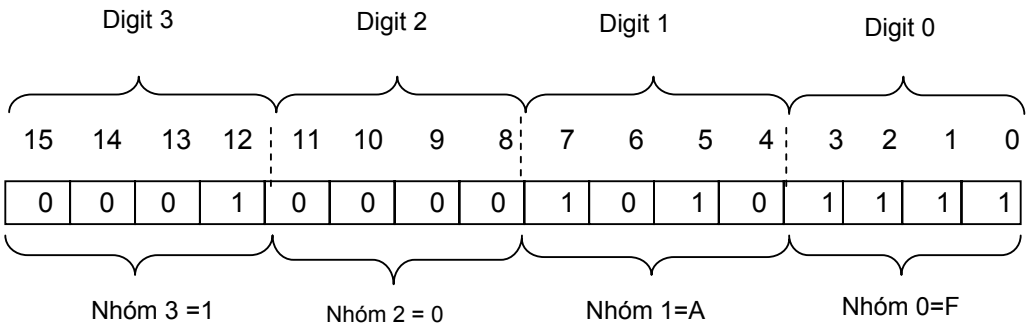


Số thập phân dưới dạng BCD :

$$(1750)_{10} = (0001011101010000)_{BCD}$$

1.2.3) Biểu diễn số nhị phân dưới dạng hexa :

Số nhị phân được biểu diễn dưới dạng hexa bằng cách nhóm 4 bit một bắt đầu từ phải qua trái và biểu diễn mỗi nhóm bit này bằng một chữ số (digit) hexa.



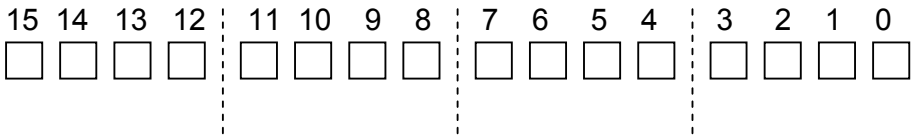
Như vậy : $0001\ \ 0000\ \ 1010\ \ 1111_2 \quad = \quad 10AF_{16}$

Chú ý :

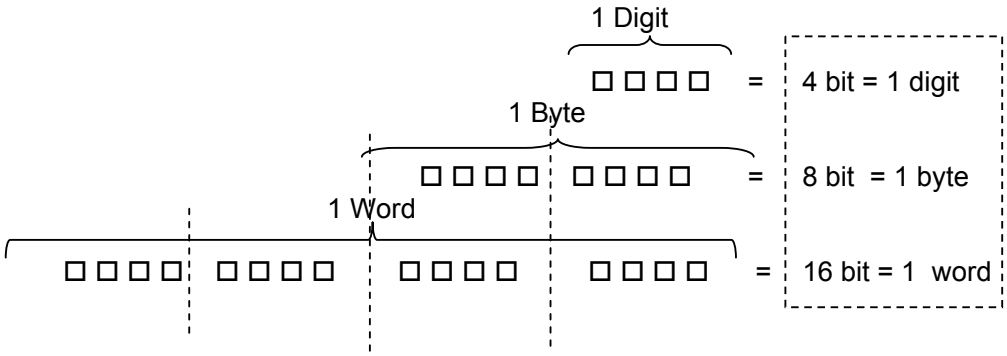
- Biểu diễn số thập phân dưới dạng hexa và BCD là không hoàn toàn tương đương nhau (cho kết quả bằng dãy số nhị phân khác nhau)
- Mã BCD được dùng chủ yếu khi đổi số thập phân ra mã nhị phân dạng BCD trong khi mã hexa được dùng phổ biến khi biểu diễn dãy số nhị phân dưới dạng ngắn gọn hơn.

1.3 Digit, Byte, Word

Dữ liệu trong PLC được mã hoá dưới dạng mã nhị phân. Mỗi chữ số được gọi là 1 bit, 8 bit liên tiếp gọi là 1 Byte, 16 bit hay 2 Byte gọi là 1 Word.



Các đại lượng liên tục (analog) như dòng điện, điện áp, ..khi ở trong PLC đều được đổi sang dạng mã nhị phân 16 bit (word) và còn được gọi là 1 kênh (Channel).



Ngoài ra để biểu diễn những số lượng lớn hơn, người ta có thêm các đơn vị sau :

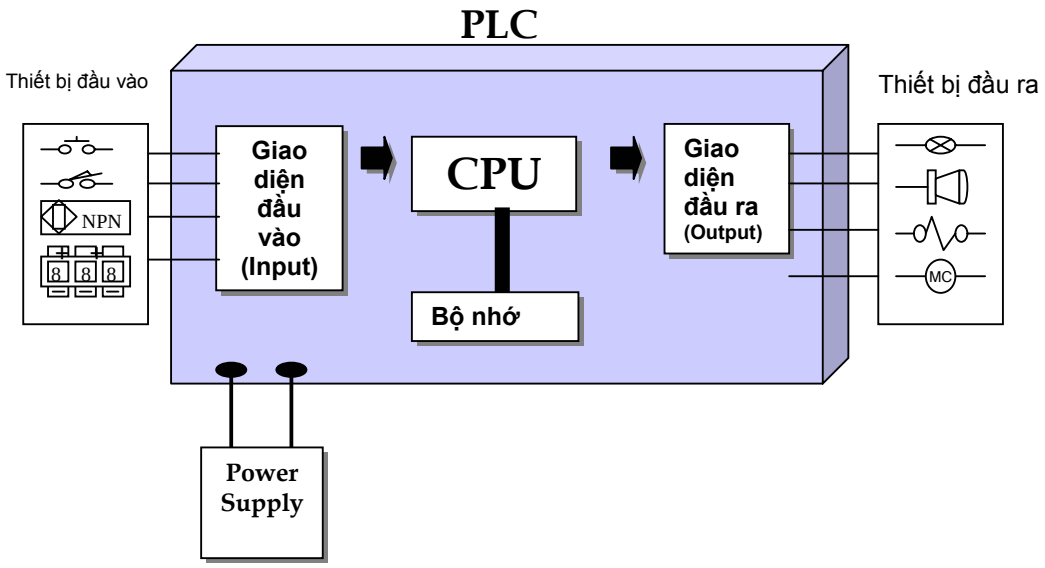
- Kilo : Trong kỹ thuật số 1 Kilobit (viết tắt là 1Kb) $=2^{10}=$ 1024 bit. Tuy nhiên để tiện tính toán người ta thường dùng là 1Kb = 1000 bit.
- Mega : 1 Mb = 1024Kb. Người ta cũng thường tính gần đúng là 1Mb=1000Kb=1.000.000 bit.
- Kilobyte và Megabyte : Tương tự như số đếm với bit nhưng các cách viết với byte là KB và MB.
- Kiloword : 1 kWord=1000 Word.
- Baud : Là cách biểu diễn tốc độ truyền tin dạng số: baud = bit/sec.

1.4 Cấu trúc của PLC (Programmable Logic Controller - gọi tắt là PLC):

Về cơ bản, PLC có thể được chia làm 5 phần chính như sau :

1. **Phần giao diện đầu vào (Input)**
2. **Phần giao diện đầu ra (Output)**
3. **Bộ xử lý trung tâm (CPU)**
4. **Bộ nhớ dữ liệu và chương trình (Memory)**
5. **Nguồn cung cấp cho hệ thống (Power Supply)**

Hình 1: Sơ đồ cấu trúc cơ bản của một bộ PLC



Nguồn cung cấp (Power Supply) biến đổi điện cung cấp từ bên ngoài thành mức thích hợp cho các mạch điện tử bên trong PLC (thông thường là 220VAC → 5VDC hoặc 12VDC).

Phần giao diện đầu vào biến đổi các đại lượng điện đầu vào thành các mức tín hiệu số (digital) và cấp vào cho CPU xử lý.

Bộ nhớ (Memory) lưu chương trình điều khiển được lập bởi người dùng và các dữ liệu khác như cờ, thanh ghi tạm, trạng thái đầu vào, lệnh điều khiển đầu ra,... Nội dung của bộ nhớ được mã hoá dưới dạng mã nhị phân.

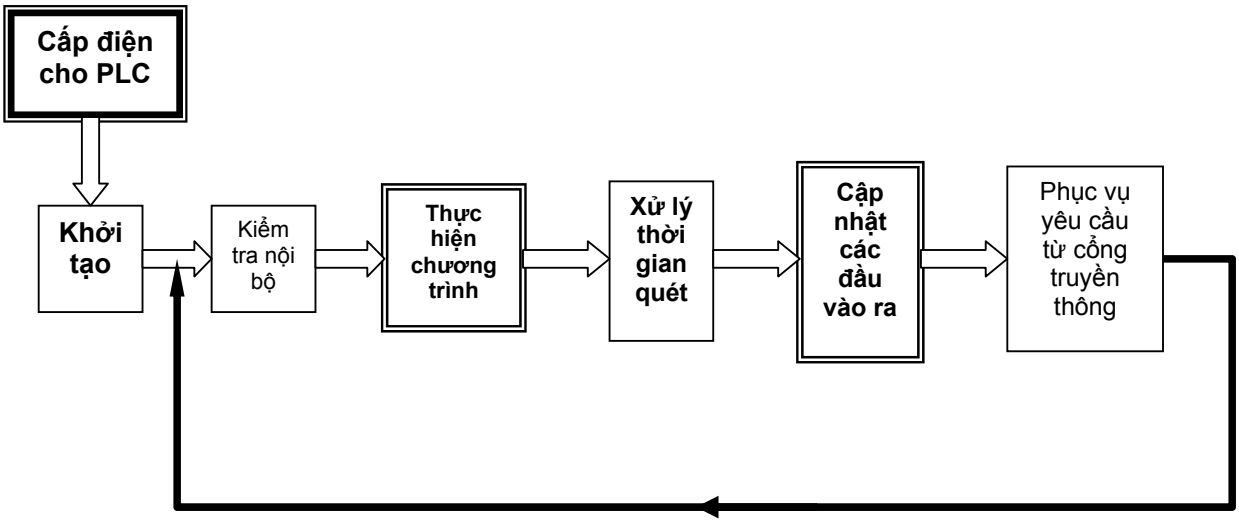
Bộ xử lý trung tâm (CPU) tuần tự thực thi các lệnh trong chương trình lưu trong bộ nhớ, xử lý các đầu vào và đưa ra kết quả kết xuất hoặc điều khiển cho phần giao diện đầu ra (output).

Phần giao diện đầu ra thực hiện biến đổi các lệnh điều khiển ở mức tín hiệu số bên trong PLC thành mức tín hiệu vật lý thích hợp bên ngoài như đóng mở rơle, biến đổi tuyến tính số-tương tự,..

Thông thường PLC có kiến trúc kiểu module hoá với các thành phần chính ở trên có thể được đặt trên một module riêng và có thể ghép với nhau tạo thành một hệ thống PLC hoàn chỉnh.
Riêng loại Micro PLC như CPM1/2(A) và CP1L/1H là loại tích hợp sẵn toàn bộ các thành phần trong một bộ.

1.5 Hoạt động của PLC

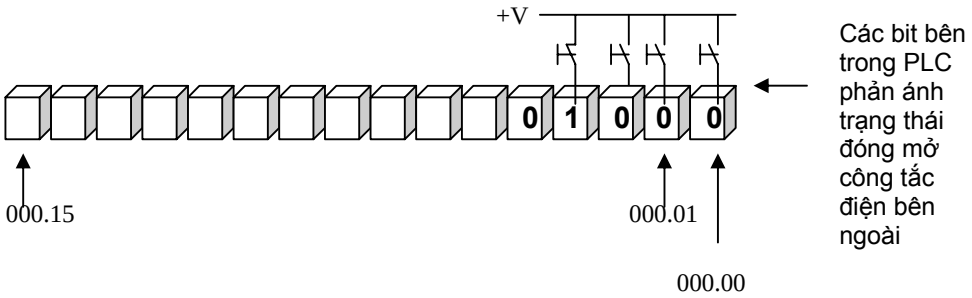
Hình 2 dưới là lưu đồ thực hiện bên trong PLC, trong đó 2 phần quan trọng nhất là **Thực hiện chương trình** và **Cập nhật đầu vào ra**. Quá trình này được thực hiện liên tục không ngừng theo một vòng kín gọi là scan hay cycle hoặc sweep. Phần thực hiện chương trình gọi là program scan chỉ bị bỏ qua khi PLC chuyển sang chế độ PROGRAM.



Hình 2: Lưu đồ thực hiện trong PLC

Về chi tiết thông số kỹ thuật của PLC loại CP1L/1H, xin tham khảo catalog và tài liệu hướng dẫn sử dụng đi kèm.

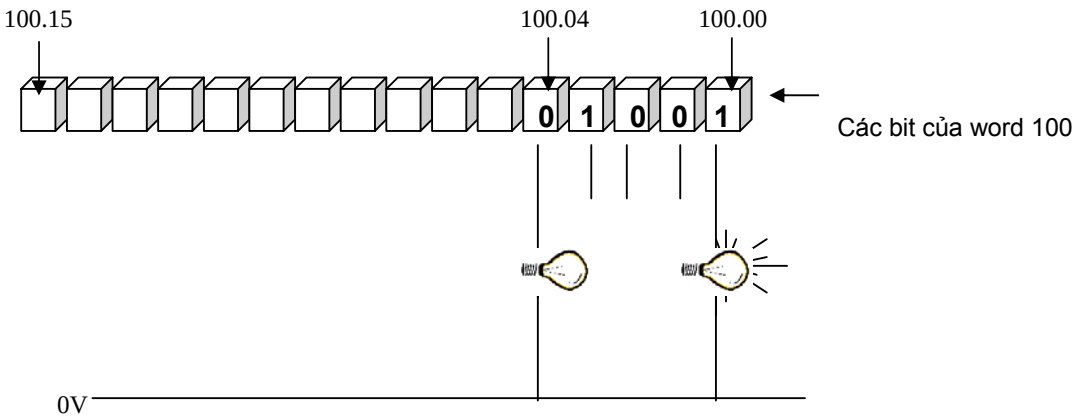
1.6 Các bit đầu vào trong PLC và các tín hiệu điện bên ngoài



Hình 3: Các bit đầu vào

Các bit trong PLC phản ánh trạng thái đóng mở của công tắc điện bên ngoài như trên hình. Khi trạng thái khoá đầu vào thay đổi (đóng/mở), trạng thái các bit tương ứng cũng thay đổi tương ứng (1/0). Các bit trong PLC được tổ chức thành từng word; ở ví dụ trên hình, các khoá đầu vào được nối tương ứng với word 000.

1.7 Các bit đầu ra trong PLC và các thiết bị điện bên ngoài



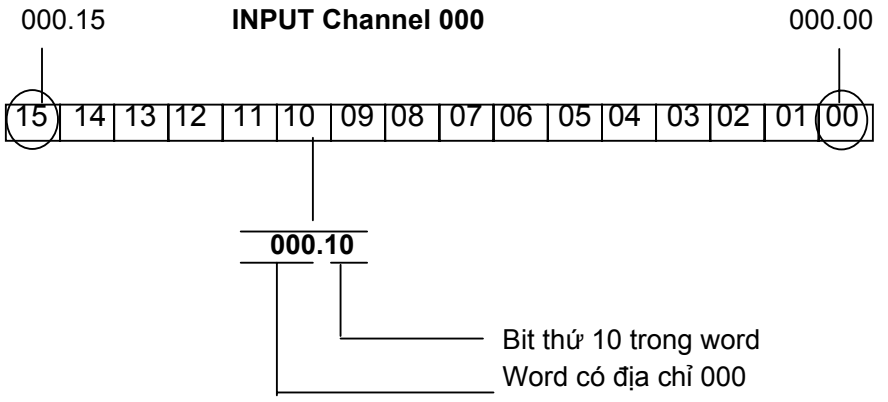
Hình 4 : Các bit đầu ra và thiết bị điện bên ngoài

Trên hình 4 là ví dụ về các bit điều khiển đầu ra của PLC. Các bit của word 0100 (từ 100.00 đến 100.15) sẽ điều khiển bật tắt các đèn tương ứng với trạng thái ("1" hoặc "0") của nó.

1.8 Các địa chỉ bộ nhớ trong CP1L/1H

Các địa chỉ dạng bit trong trong PLC được biểu diễn dưới dạng như sau :

[Tiền tố][Địa chỉ word] . [Số của bit trong word]



Trong đó tiền tố là ký hiệu của loại địa chỉ bộ nhớ. Ví dụ : SR cho Special Relay, LR cho Link Relay, IR cho Internal Relay,... Riêng vùng nhớ Internal Relay và CIO là các bit vào ra I/O không cần có tiền tố IR hay CIO khi tham chiếu. Special Relay cũng thường được coi là Internal Relay và không cần có tiền tố.

Ví dụ :

000.00 là bit thứ nhất của word 000
000.01 là bit thứ hai của word 000

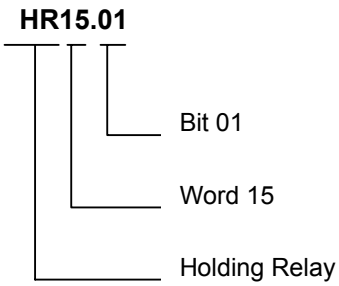
. . . . . .

000.15 là bit thứ 16 của word 000

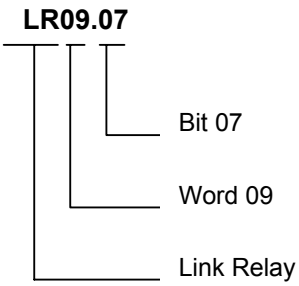
Chú ý : Dấu chấm phân cách giữa địa chỉ word và bit đổi khi có thể được bỏ đi; nhưng khi nhập thì dấu chấm vẫn nên phải nhập vào để tránh nhầm lẫn.

Sau đây là ví dụ về 2 trong số những bộ nhớ đặc biệt trong PLC của OMRON

Holding Relay



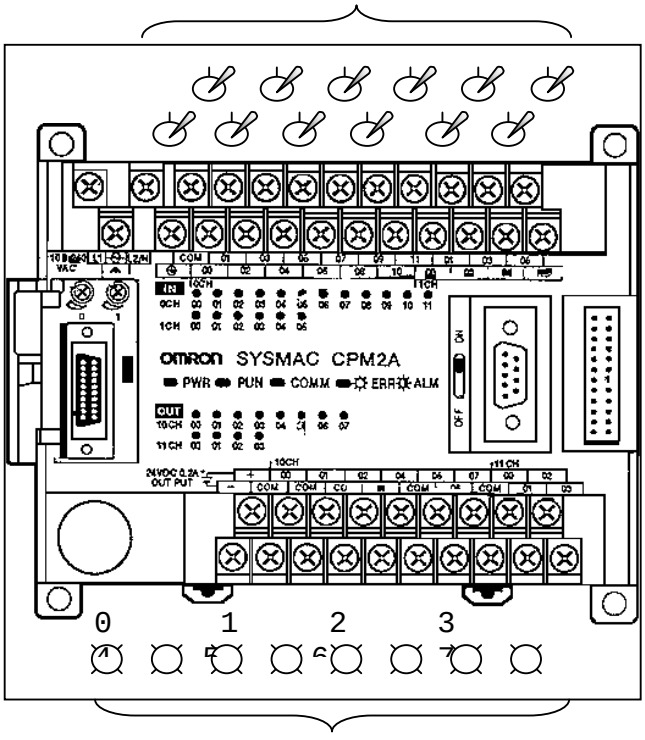
Link Relay



Phần II: Làm quen với PLC

1.9 Giới thiệu về bộ training kit CP1L/1H

A) C¸c kho¸ chuyÓn m¹ch ®Çu vµo (INPUT SWITCHES)

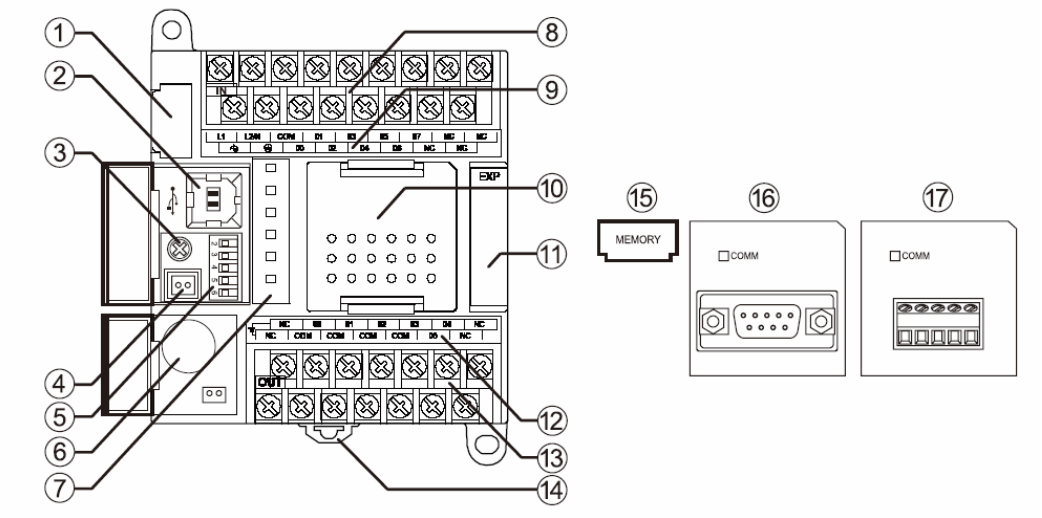


B) Các đèn chỉ thị trạng thái đầu ra (OUTPUT INDICATORS)

Hình 1 : **Bộ Training CP1L/1H**

Bộ CP1L/1H dành cho việc đào tạo (CP1L/1H Training kit) là một bộ điều khiển lập trình loại nhỏ loại CP1L-L14 có thêm 8 khoá chuyển mạch đầu vào để mô phỏng các đầu vào số (đánh số từ 0 đến 7) và có sẵn 6 đèn chỉ thị trạng thái đầu ra (đánh số từ 00 đến 05) được điều khiển bởi chương trình do người dùng lập (User program).

1.9.1 Các thành phần trên bộ CP1L-14 :



Hình 2

Các thành phần chính trên bộ CP1L/1H trên hình :

1. Khe cắm card nhớ (Memory cassette)
   Dùng để gắn card nhớ (15) để lưu chương trình, các thông số & bộ nhớ dữ liệu của CP1L/1H. Nó cũng có thể dùng để copy & nạp chương trình sang các bộ PLC loại CP1L/1H khác mà không cần dùng máy tính
2. Peripheral USB port
   Dùng để nối với máy tính cho việc lập trình
3. Núm chiết áp chỉnh (Analog adjuster)
   Khi quay chiết áp này, giá trị của bộ nhớ trong PLC ở địa chỉ A642 sẽ thay đổi trong khỏang 0-255.
4. Đầu nối đầu vào chiết áp analog
   Đầu nối này dùng kết nối với tín hiệu đầu vào từ 0-10VDC, để thay đổi giá trị của thanh ghi bộ nhớ A643 trong khỏang 0-255. Đầu vào này không có cách ly.
5. DIP switch
   Dùng để đặt các thông số hoạt động như cấm ghi vào vùng nhớ chương trình, tự động nạp dự liệu từ card nhớ,..
6. Pin
   Lưu nội dung RAM & đồng hồ khi nguồn tắt
7. Các đèn báo hoạt động
   Xem bảng dưới
8. - Dây nguồn điện cung cấp cho PLC (Power Supply Input Terminal)
   - Đầu nối đất tín hiệu (Functional Earth Terminal) (chỉ đối với loại AC) nhằm tăng khả năng chống nhiễu và tránh điện giật
   - Đầu nối đất bảo vệ (Protective Earth Terminal) để tránh điện giật.
   PLC có thể được cung cấp bằng nguồn điện xoay chiều 100-240VAC hoặc 1 chiều 24VDC (tuỳ loại).
   - Đầu nối tín hiệu vào (Input Terminal)
9. Các đèn chỉ thị trạng thái đầu vào (Input Indicator)
   Đèn LED trong nhóm này sẽ sáng khi đầu vào tương ứng lên ON
10. Khe cắm các card truyền thông mở rộng tùy chọn
    Dùng để cắm thêm các card RS-232C (16) hay RS-422A/485 (17).
    Model với 14/20 I/O có 1 khe cắm có thể lắp được 1 card. Model 30/40/60 I/O có 2 khe cắm có thể lắp được 2 card truyền thông mở rộng
11. Đầu nối với module vào ra mở rộng (Expansion I/O Unit)

Dùng để nối module có CPU (là module chính có bộ xử lý trung tâm - CPU và chứa chương trình ứng dụng - User program) với module vào ra mở rộng (Expansion I/O Unit) để bổ sung đầu vào ra cho module chính

12. Các đèn chỉ thị trạng thái đầu ra (Output Indicator)
Đèn LED trong nhóm này sẽ sáng khi đầu ra tương ứng lên ON
13. Đầu nối nguồn cấp DC ra từ PLC (DC Power Supply Output Terminal) & đầu nối cho đầu ra
Điện áp ra ở đầu nối nguồn cấp DC chuẩn là 24VDC với dòng định mức là 0,3A có thể được dùng cấp cho các đầu vào số DC
14. Chốt gắn trên thanh ray DIN
15. Card nhớ (Memory cassette) (tùy chọn)
Dùng để lưu dữ liệu từ bộ nhớ flash trong CPU. Cắm vào khe cắm Card nhớ (1).
16. Card truyền thông RS-232C (tùy chọn)
Cắm vào khe cắm truyền thông (10).
17. Card truyền thông RS-422A/485 (tùy chọn)
Cắm vào khe cắm truyền thông (10).

Các đèn LED chỉ thị trạng thái của PLC (PLC Status Indicators)

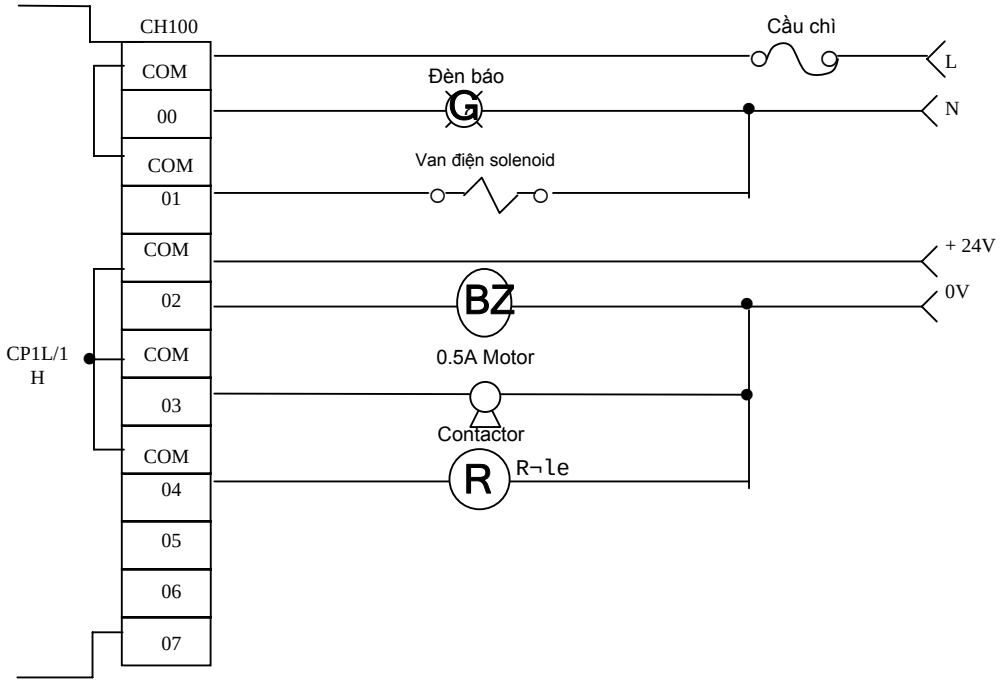
Đèn	Trạng thái	Chức năng
POWER (màu xanh)	Bật	PLC đang được cấp điện bình thường
	Tắt	PLC không được cấp điện bình thường (không có điện, điện yếu,..)
RUN (màu xanh)	Bật	PLC đang hoạt động ở chế độ RUN hay MONITOR.
	Tắt	PLC đang ở chế độ PROGRAM hoặc bị dừng
ERR/ALM (Đỏ)	Sáng	PLC gặp lỗi nghiêm trọng (chương trình PLC ngừng chạy), bao gồm cả lỗi FALS hay lỗi phần cứng (WDT). Tất cả các đầu ra sẽ tắt
	Nhấp nháy	PLC gặp một lỗi không nghiêm trọng (PLC tiếp tục chạy ở chế độ RUN)
	Tắt	PLC hoạt động bình thường không có lỗi
PRPHL (Vàng)	Sáng	Đang truyền thông qua cổng USB
	Tắt	Hiện không có truyền thông qua cổng USB
INH (Vàng)	Sáng	Bit tắt đầu ra (A500.15) bật. Lúc này tất cả các đầu ra trên PLC sẽ tắt, bất kể chương trình điều khiển
	Tắt	Hoạt động như bình thường
BKUP (Vàng)	Sáng	• Chương trình, thông số hay bộ nhớ dữ liệu đang được ghi vào bộ nhớ flash hay card nhớ. • Chương trình, thông số hay bộ nhớ dữ liệu đang được đọc lại từ bộ nhớ ngòai sau khi bật điện Lưu ý: không tắt điện trong khi đèn này đang sáng
	Tắt	Hoạt động như bình thường

Khi gặp một sự cố trầm trọng, các đèn chỉ thị trạng thái đầu vào sẽ thay đổi như sau :

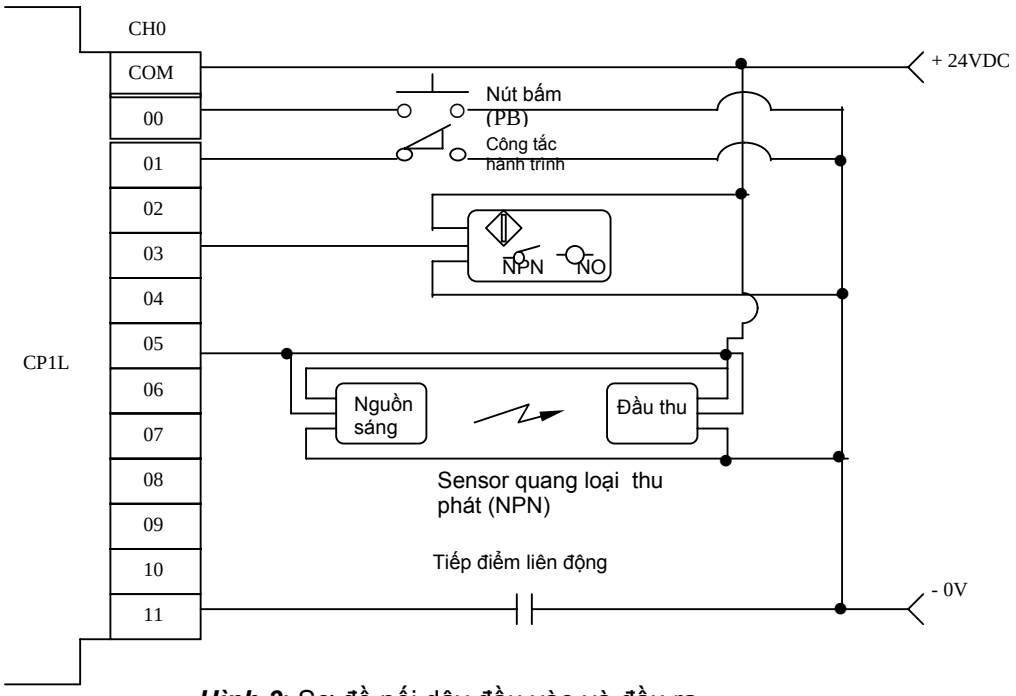
- Khi có lỗi CPU hay lỗi với bus vào/ ra (CPU Error/ I/O Bus Error) : các LED đầu vào sẽ tắt.
- Khi có lỗi với bộ nhớ hoặc lỗi hệ thống (Memory Error/ System Error) : các LED đầu vào vẫn giữ trạng thái của chúng trước khi xảy ra lỗi cho dù trạng thái thực đầu vào đã thay đổi.

1.9.2 Ví du về đấu dây (CP1L-20)

a/ Nối dây đầu ra (loại tiếp điểm rơle) :



b/ Nối dây đầu vào (24VDC) :



Hình 3: Sơ đồ nối dây đầu vào và đầu ra

1.9.3 Định địa chỉ bộ nhớ các đầu vào ra (I/O ALLOCATION - IR BIT)

Các đầu vào ra (I/O) trên PLC đều được định (assign) một địa chỉ bộ nhớ xác định trong vùng nhớ IR để tham chiếu trong chương trình. Các đầu nối vào ra này được đánh số sẵn và được định địa chỉ theo bảng dưới đây.
Trên bảng 2 là địa chỉ bộ nhớ của các loại PLC họ CP1L/1H.

Bảng 2 Địa chỉ bộ nhớ vào ra của các loại PLC họ CP1L/1H (14,20,30,40,60 I/O)

Số lượng đầu vào ra trên module CPU	Đầu nối trên module CPU	
	Input	Output
14	8 đầu: 000.00 đến 000.07	6 đầu: 100.00 đến 100.05
20	12 đầu: 000.00 đến 000.11	8 đầu: 100.00 đến 100.07
30	18 đầu: 000.00 đến 000.11 001.00 đến 001.05	12 đầu: 100.00 đến 100.07 101.00 đến 101.03
40	24 đầu 000.00 đến 000.11 và 001.00 đến 001.11	16 đầu 100.00 đến 100.07 101.00 đến 101.07
60	36 đầu	24 đầu

	000.00 đến 000.11 và 001.00 đến 001.11 002.00 đến 002.11	100.00 đến 100.07 và 101.00 đến 101.07 và 102.00 đến 102.07

Địa chỉ bộ nhớ trên module mở rộng

Word trên module mở rộng sẽ sử dụng word tiếp theo của vùng nhớ input hay output tương ứng chưa sử dụng bởi module mở rộng trước đó hoặc module CPU.

CPU unit	Các word dành sẵn		Số module mở rộng được phép nối
	Vùng Input	Vùng Output	
10-point I/O unit	0 CH	100 CH	0
14-point I/O unit	0 CH	100 CH	1
20-point I/O unit	0 CH	100 CH	1
30-point I/O unit	0 CH, 1 CH	100 CH, 101 CH	3
40-point I/O unit	0 CH, 1 CH	100 CH, 101 CH	3
60-point I/O unit	0 CH, 1 CH, 2CH	100 CH, 101 CH, 102 CH	3

Bảng 3: Các loại module mở rộng loại CPM1A của họ CP1L/1H

Loại	Số đầu vào	Số đầu ra	Loại đầu vào ra	Mã
20 đầu vào ra I/O	12	8	Rơle	CPM1A-20EDR
			Transistor NPN	CPM1A-20EDT
			Transistor PNP	CPM1A-20EDT1
8 đầu vào	8	0		CPM1A-8ED
8 đầu ra	0	8	Rơle	CPM1A-8ER
			Transistor NPN	CPM1A-8ET
			Transistor PNP	CPM1A-8ET1
Đầu vào ra analog	2	1	Analog	CPM1A-MAD01
Module vào ra Slave Compobus/S	8	8		CPM1A-SRT2I
Module đầu vào nhiệt độ	2 hoặc 4	0	Cặp nhiệt	CPM1A-TS001/TS002
			Nhiệt điện trở	CPM1A-TS101/TS102

Các module mở rộng đặc biệt

Module	Model	Thông số			Khối lượng
Analog I/O Units	CPM1A-MAD01	2 analog inputs	0 đến 10 V/1 đến 5 V/4 đến 20 mA	Độ phân giải: 256	150 g max.
		1 analog output	0 đến 10 V/−10 đến +10 V/4 đến 20 mA		
	CP1W-MAD11 CPM1A-MAD11	2 analog inputs	0 đến 5 V/1 đến 5 V/0 đến 10 V/−10 đến +10 V/0 đến 20 mA/4 đến 20 mA	Độ phân giải: 6.000	

		1 analog output	1 đến 5/0 đến 10 V/−10 đến +10 V/0 đến 20 mA/4 đến 20 mA		
Analog Input Units	CP1W-AD041 CPM1A-AD041	4 analog inputs	0 đến 5 V/1 đến 5 V/0 đến 10 V/−10 đến +10 V/ 0 đến 20 mA/ 4 đến 20 mA	Độ phân giải: 6.000	200 g max.
Analog Output Units	CP1W-DA041 CPM1A-DA041	4 analog outputs	1 đến 5 V/0 đến 10 V/ −10 đến +10 V/ 0 đến 20 mA/ 4 đến 20 mA		
Temperature Sensor Units	CP1W-TS001 CPM1A-TS001	2 inputs	Thermocouple input K, J		250 g max.
	CP1W-TS002 CPM1A-TS002	4 inputs			
	CP1W-TS101 CPM1A-TS101	2 inputs	Platinum resistance thermometer input Pt100, JPt100		
	CP1W-TS102 CPM1A-TS102	4 inputs			
DeviceNet I/O Link Unit	CPM1A-DRT21	As a DeviceNet Slave, 32 inputs & 32 outputs are allocated.			200 g max.
CompoBus/S I/O Link Unit	CP1W-SRT21 CPM1A-SRT21	As a CompoBus/S slave, 8 inputs & 8 outputs are allocated.			200 g max.

Các địa chỉ trên module mở rộng loại đầu vào/ra số:

Module			**Bit đầu vào**			**Bit đầu ra**		
			Số lượng bit	**Số lượng words**	**Địa chỉ**	**Số lượng bit**	**Số lượng words**	**Địa chỉ**
Module với 8 input		CP1W-8ED CPM1A-8ED	8 bit	1 word	CIO m (bit 00 đến 07)	---	Không có	Không có
Module với 8 output	Relays	CP1W-8ER CPM1A-8ER	---	Không có	Không có	8 bit	1 word	CIO n (bit 00 đến 07)
	Sinking transistors	CP1W-8ET CPM1A-8ET	---	Không có	Không có	8 bit	1 word	CIO n (bit 00 đến 07)
	Sourcing transistors	CP1W-8ET1 CPM1A-8ET1	---	Không có	Không có	8 bit	1 word	CIO n (bit 00 đến 07)
Module với 16 relay outputs		CP1W-16ER CPM1A-16ER	---	Không có	Không có	16 bit	2 words	CIO n (bit 00 đến 07) CIO n+1 (bit 00 đến 07)
Module với 20 I/O	Relay	CP1W-20EDR1 CPM1A-20EDR1	12 bit	1 word	CIO m (bit 00 đến 11)	8 bit	1 word	CIO n (bit 00 đến 07)
	Sinking transistors	CP1W-20EDT CPM1A-20EDT	12 bit	1 word	CIO m (bit 00 đến 11)	8 bit	1 word	CIO n (bit 00 đến 07)
	Sourcing transistors	CP1W-20EDT1 CPM1A-20EDT1	12 bit	1 word	CIO m (bit 00 đến 11)	8 bit	1 word	CIO n (bit 00 đến 07)
Module với 40 I/O	Relays	CP1W-40EDR CPM1A-40EDR	24 bit	2 words	CIO m (bit 00 đến 11) CIO m+1 (bit 00 đến 11)	16 bit	2 words	CIO n (bit 00 đến 07) CIO n+1 (bit 00 đến 07)
	Sinking transistors	CP1W-40EDT CPM1A-40EDT	24 bit	2 words	CIO m (bit 00 đến 11) CIO m+1 (bit 00 đến 11)	16 bit	2 words	CIO n (bit 00 đến 07) CIO n+1 (bit 00 đến 07)
	Sourcing transistors	CP1W-40EDT1 CPM1A-40EDT1	24 bit	2 words	CIO m (bit 00 đến 11) CIO m+1 (bit 00 đến 11)	16 bit	2 words	CIO n (bit 00 đến 07) CIO n+1 (bit 00 đến 07)

Trong đó:

- m là ký hiệu của word đầu vào mở rộng
- n là ký hiệu của word đầu ra mở rộng

Ví dụ: Với bộ CP1L/1H-30CDR-A với 30 đầu vào/ra thì:

- Trên CPU Unit: Input chiếm các word 000 và 001
  Output chiếm các word 100 và 101

- Nếu nối thêm module mở rộng CP1A-20EDR (12 vào/8 ra) thì :
  Input chiếm word 002, các bit từ 00 đến 11
  Output chiếm word 102, các bit từ 00 đến 07
- Nếu nối thêm tiếp module mở rộng CP1W-20EDT (12 vào/8 ra) thì:
  - Input chiếm word 003, các bit từ 00 đến 11
  - Output chiếm word 103, các bit từ 00 đến 07
- Nếu nối thêm tiếp module mở rộng CP1W-8ED (8 vào) thì:
  - Input chiếm word 004, các bit từ 00 đến 07
  - Không có output word cho module này

Các word còn lại nếu chưa nối thêm module mở rộng nào khác sẽ là tự do cho chương trình sử dụng

Về các module khác, xin tham khảo tài liệu đi kèm của các module này, catalog hoặc cuốn “Operation Manual” & "Programming Manual".

1.9.4 Các vùng nhớ trong CP1L/1H

Bộ nhớ trong PLC được chia thành các vùng (area) khác nhau với các chức năng riêng biệt như sau:

Vùng nhớ (area)		Words		Bit	
			Ở phần mềm CX-P		Ở phần mềm CX-P
CIO area	I/O area	00 đến 199	0 đến 199	00000 đến 19915	0.00 đến 199.15
	1:1 link area	3000 đến 3063 CH	3000 đến 3063	300000 đến 306300	3000.00 đến 3063.00
	Serial PLC link area	3100 đến 3189 CH	3100 đến 3189	310000 đến 318915	3100.00 đến 3189.15
	Work area	3800 đến 6143 CH	3800 đến 6143	380000 đến 614300	3800.00 đến 6143.00
Work area		W000 đến W511 CH	W000 đến W511	W00000 đến W51115	W0.00 đến W511.15
Holding area		H000 đến H511 CH	H000 đến H511	H00000 đến H51115	H0.00 đến H511.15
Auxiliary area		A000 đến A959 CH	A000 đến A959	A00000 đến A95915	A0.00 đến A959.15
DM area		D00000 đến D32767*	D0 đến D32767*	-	-
Timer		T000 đến T511	T0 đến T511	T000 đến T511	T0000 đến T0511
Counter		C000 đến C511	C000 đến C511	C000 đến C511	C0000 đến C0511

*Đối với loại 14/20 I/O: D0 - D9999, D32000 - D32767.

Chức năng các vùng nhớ:

Vùng nhớ		Chức năng
CIO area	Input area	Các bit này có thể được gán cho các đầu dây vào ra I/O.
	Output area	
	1:1 link area	Dùng cho kết nối 1:1 với 1 PLC khác.
	Serial PLC link area	Dùng cho kết nối 1:n với 1 PLC khác.
	Work area	Work bit có thể được sử dụng tuỳ ý trong chương trình
SR area		Các bit này phục vụ cho các chức năng riêng biệt như cờ báo và các bit điều khiển.
TR area		Các bit này lưu dữ liệu và lưu trạng thái ON/OFF tạm thời tại các nhánh rẽ chương trình.
HR area[2]		Các bit này lưu dữ liệu và lưu lại trạng thái ON/OFF của chúng khi ngắt điện.

AR area[2]		Các bit này phục vụ cho các chức năng riêng biệt như cờ báo và các bit điều khiển.
Timer/Counter area		Các số này có thể được dùng cho cả timers và counters.
DM area	Read/Write[2]	Dữ liệu lưu ở vùng bộ nhớ DM chỉ có thể được truy cập theo word. Giá trị của các word tự lưu giá trị khi mất điện.
	Error log[4]	Dùng để lưu thời gian xuất hiện và mã của lỗi. Các word này có thể được dùng như là các word DM đọc/ghi thông thường khi chức năng lưu lỗi hiện không được sử dụng.
	Read-only[4]	Chương trình không thể ghi đè lên các word này
	PC Setup[4]	Dùng lưu các thông số khác nhau điều khiển hoạt động của PLC.

Ghi chú :

1. Các bit CIO Area và LR khi không được dùng cho các chức năng đã định của chúng có thể được dùng như bit tự do trong chương trình (work bit).
2. Nội dung của các thanh ghi HR, LR, counter, và vùng bộ nhớ DM đọc/ghi được nuôi bằng pin. Ở nhiệt độ 25^0C, pin có thể lưu nội dung bộ nhớ trong vòng 5 năm.
3. Khi truy cập giá trị hiện hành (PV) của timer và counter, các số của timer và counter (ví dụ C001, T005) được dùng như là các dữ liệu dạng word; khi truy cập bit cờ báo kết thúc (Completion Flag) của timer và counter, chúng được dùng như là các bit trạng thái.
4. Dữ liệu ở các thanh ghi từ DM6144 đến DM6655 không thể bị ghi đè bởi chương trình nhưng chúng có thể được thay đổi từ thiết bị ngoại vi.

Các ký hiệu hằng số:

Ký hiệu	**Nội dung/mục đích**
#0000 đến 9999 (BCD)	Các giá trị của Timer/counter, Lệnh số học BCD,..
#0000 đến FFFF (Hex)	Giá trị so sánh cho các lệnh so sánh, copy dữ liệu, Lệnh số học BIN,..
&0 đến 65535	Ký hiệu số thập phân không dấu Chỉ có 1 số lệnh đặt biệt dùng kiểu dữ liệu này.

1.9.5 Các cờ báo

Các cờ báo trong PLC được CPU tự động đặt để phản ánh các trạng thái & giá trị của hoạt động bên trong PLC hoặc của chương trình.

Tên	**Nhãn** (symbol)	**Ở phần mềm CX-P**	**Chức năng**
Cờ báo lỗi-Error flag	ER	P_ER	• Bật ON khi lệnh dùng dữ liệu BCD muốn sử dụng dữ liêu không phải ở dạng BCD. • Bật ON khi tham số của lệnh không hợp lệ (ví dụ giá trị vượt ra ngòai khỏang).
Cờ báo lỗi truy cập-Access error flag	AER	P_AER	Bật ON khi truy cập vào vùng nhớ không được phép
Cờ nhớ-Carry flag	CY	P_CY	• Bật ON khi số lượng digit tăng hay giảm khi thực hiện lệnh số học.

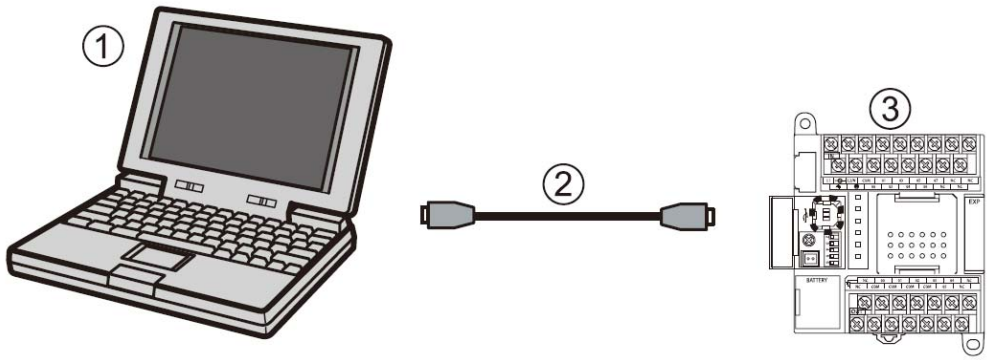
			• Các lệnh dịch dự liệu & số học có thể dùng cờ này như 1 phần của quá trình thực hiện
Cờ bằng-Equals flag	=	P_EQ	• Bật ON khi lệnh so sánh cho kết quả "Bằng". • Bật ON khi kết quả thực hiện bằng 0 với các lệnh tính tóan hay copy dữ liệu
Cờ không bằng-Unequal flag	< >	P_NE	Bật ON khi lệnh so sánh cho kết quả "Không Bằng".
Cờ lớn hơn-Greater than flag	>	P_GT	Bật ON khi lệnh so sánh cho kết quả "Tham số 1 > Tham số 2".
Cờ lớn hơn hay bằng - Greater than or equals flag	>=	P_GE	Bật ON khi lệnh so sánh cho kết quả "Tham số 1 >= Tham số 2".
Cờ nhỏ hơn- Less than flag	<	P_LT	Bật ON khi lệnh so sánh cho kết quả "Tham số 1 < Tham số 2".
Cờ Cờ nhỏ hơn hay bằng-Less than or equals flag	<=	P_LE	Bật ON khi lệnh so sánh cho kết quả "Tham số 1 <= Tham số 2".
Cờ âm- Negative flag	N	P_N	Bật ON khi lệnh tính tóan cho kết quả byte cao =1
Cờ tràn trên-Overflow flag	OF	P_OF	Bật ON khi lệnh tính tóan cho kết quả tràn trên
Cờ tràn dưới-Underflow flag	UF	P_UF	Bật ON khi lệnh tính tóan cho kết quả tràn dưới
Cờ luôn ON- Always ON flag	ON	P_ON	Luôn luôn ON
Cờ luôn OFF- Always OFF flag	OFF	P_OFF	Luôn luôn OFF

Lưu ý:

- Các cờ báo trên khi nhập vào để sử dụng trong chương trình chỉ sử dụng tên nhãn (symbol) mà không dùng địa chỉ. Trong phần mềm CX-Programmer, các tên nhãn này bắt đầu bằng "P_", ví dụ P_OFF
- Các cờ báo trên được dùng chung cho tòan bộ chương trình, kể cả chương trình con, task,... Vì vậy, để phản ánh đúng kết quả của lệnh, cần sử dụng các cờ này ngay sau các lệnh tác động lên các cờ báo.

1.10 Nối ghép giữa PLC và thiết bị ngoại vi :

Để PLC có thể giao tiếp được với thiết bị ngoại vi qua cổng USB Peripheral Port, chỉ cần 1 cáp nối USB thông thường (2).



Khi nối bằng cáp USB chỉ cho phép kết nối 1 máy tính với 1 PLC. Không nên rút cáp nối ra khỏi PLC hay máy tính trong khi đang online, nếu không sẽ có thể bị lỗi sau:
[Windows 2000, XP]

Nếu cắm lại cáp USB thì CX-Programmer vẫn chưa thể về trạng thái online với PLC. Đầu tiên cần chuyển CX-Programmer về trạng thái offline, cắm lại cáp USB, rồi chuyển CX-Programmer về trạng thái online.
[Windows 98, Me]
Nếu CX-Programmer ở trạng thái online mà rút dây USB ra, hệ thống có thể bị lỗi gây treo máy & cần khởi động lại máy tính.

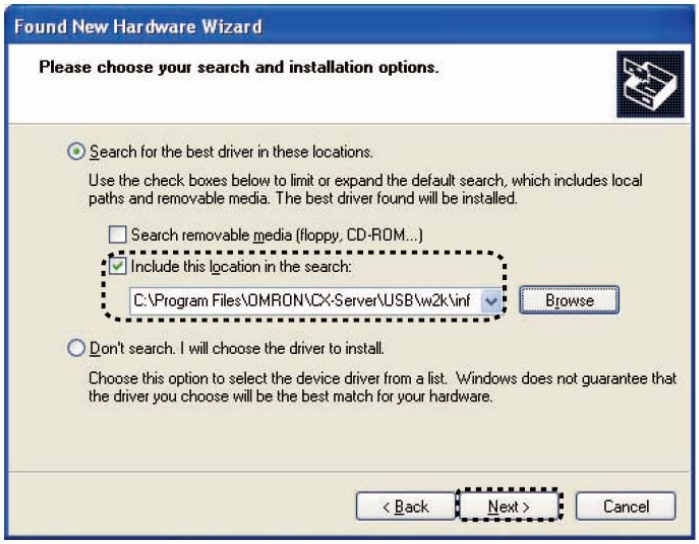
1.11 Các bước thực hiện cài đặt USB driver cho PLC & kết nối PLC với máy tính

1- Bật PC & PLC
2- Nối PC & PLC với nhau qua cáp USB
Máy tính sẽ nhận diện thiết bị được kết nối là OMRON-PLC.

Hộp thoại Found New Hardware Wizard sẽ hiển thị các bước cài đặt USB driver
3- **Ở hộp thoại Found New Hardware Wizard, chọn [No, not this time] rồi bấm Next**

4- Chọn **[Install from a list of specific location (Advanced)]**, rồi bấm [Next].

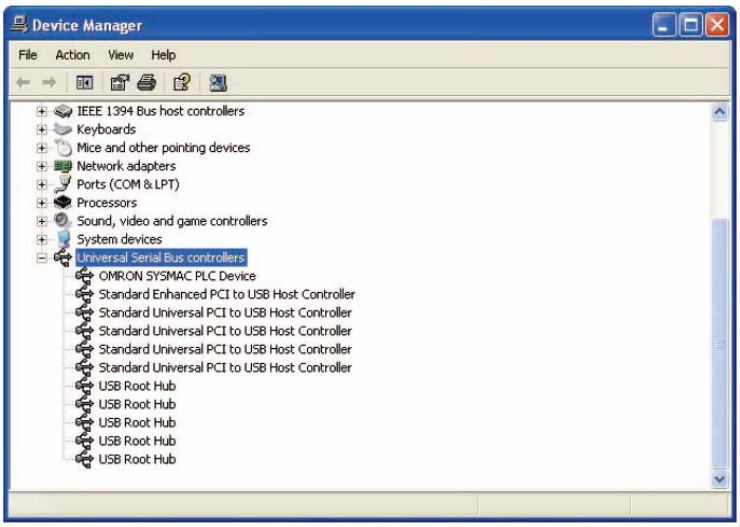
5- Chọn [Include this location in the search] và kiểm tra đường dẫn [C:\Program Files\OMRON\CX-Server\USB\Win2000_XP\Inf] rồi bấm Next



Quá trình cài đặt USB driver sẽ được thực hiện.
Bấm chọn [Continue Anyway] khi hộp thoại cảnh báo hiện ra:

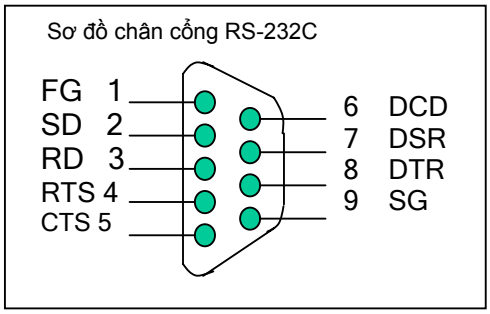
6- **Bấm [Finish] để kết thúc việc cài đặt USB driver**

Để kiểm tra USB driver đã được cài đúng chưa, bấm tổ hợp phím [Windows] + Break để hiển thị hộp thoại System properties (hoặc bấm Start, chọn Settings, chọn Control Panel, rồi chọn System), chọn **Hardware** rồi chọn **[Device Manager]**,
USB driver **[OMRON SYSMAC PLC Device]** sẽ được hiển thị ở phần **[Universal Serial Bus controllers].**



Nếu không thấy **[OMRON SYSMAC PLC Device]** thì cần cài đặt lại USB driver.

Nếu nối thiết bị RS-232C bên ngoài với PLC qua cổng RS-232C trên card truyền thông cắm thêm trên CPU Unit, chỉ cần có 1 cáp RS-232C.



Hình 4 : Sơ đồ chân cổng RS-232C trên card truyền thông cắm thêm

1.12 Các tính năng chính của bô CP1L

1.12.1) Module CP1L chính cung cấp 6 loại với số lượng I/O khác nhau : 10, 14, 20, 30, 40 và 60 I/O. Tất cả đều có sẵn cổng USB.

1.12.2) Có thể lắp thêm tối đa là 1 (với loại 14 & 20 I/O) hoặc 3 module mở rộng (với loại 30, 40 & 60 I/O) (xem bảng 3)

1.12.3) Input time constant : để giảm ảnh hưởng do nhiễu hay do tín hiệu vào lập bập không ổn định, đầu vào của CP1L/1H có thể được đặt một hằng số thời gian trễ là 1, 2, 4, 8, 16, 32, 64 hay 128 ms.

1.12.4) Lập trình bằng ngôn ngữ bậc thang (ladder), dòng lệnh (statement list), lệnh có cấu trúc (Structured text), Khối lệnh (Function Block) hoặc lưu đồ (SFC) bằng phần mềm chạy trong Windows với CX-Programmer. Không hỗ trợ bộ lập trình cầm tay.

1.12.5) Có 2 chiết áp chỉnh độ lớn thanh ghi bên trong PLC (Analog Volume Adjustment) với khoảng thay đổi giá trị từ 0-250 (BCD) thích hợp cho việc chỉnh định timer hoặc counter bằng tay.

1.12.6) Có thể nhận xung vào từ Encoder với 2 chế độ chính :
- Incremental mode . . . 100 KHz
- UP/DOWN mode . . . 50 KHz

1.12.7) Có Interval Timer tốc độ cao với thời gian đặt từ 0.5 ms - 319.968 ms. Timer có thể được đặt để kích hoạt ngắt đơn (One-shot Interrupt) hoặc lặp đi lặp lại ngắt theo định kỳ (scheduled interrupt).

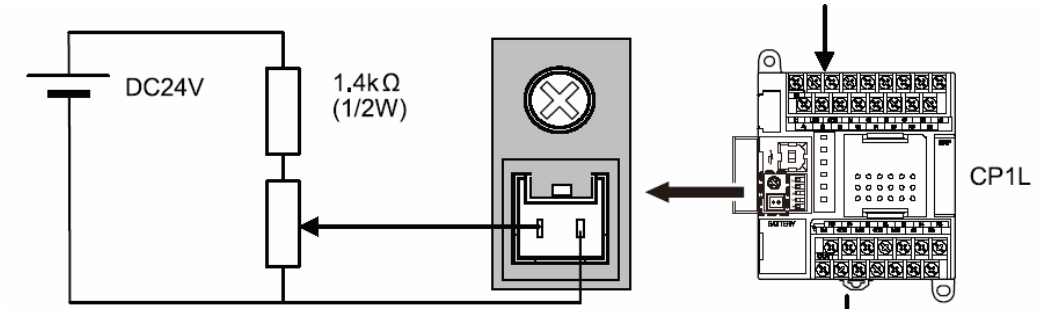
1.12.8) Có đầu vào tốc độ cao để phát hiện các tín hiệu với độ rộng xung ngắn (tới 50 microsec) không phụ thuộc vào thời gian quét chương trình.

1.12.9) Truyền thông theo chuẩn Host Link/NT Link hoặc 1:1 Data Link qua cổng RS-232C/RS422/485 trên board cắm thêm trên CPU Unit.

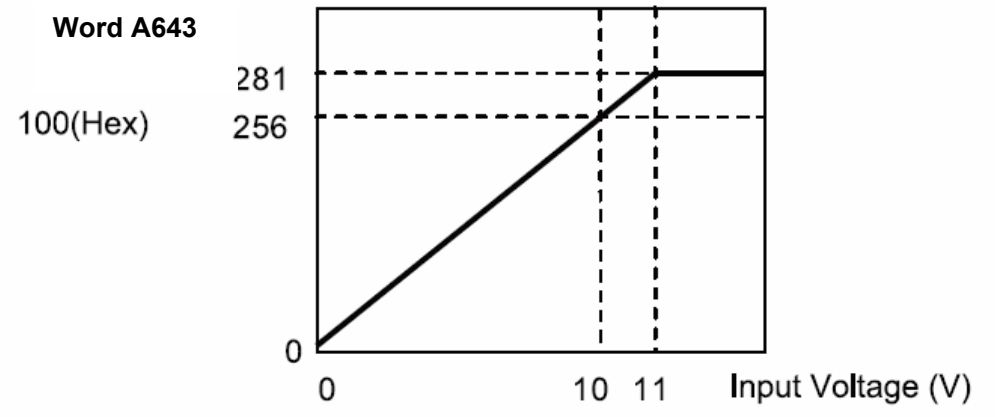
1.13 Analog Setting Function

Bộ CP1L/1H có sẵn 1 chiết áp đầu vào & 1 đầu nối chiết áp ngòai để chỉnh giá trị thanh ghi bên trong PLC (Analog Adjuster) với độ phân giải 8 bit và khoảng giá trị thay đổi từ 0-255 (BCD).

Chiết áp Analog Adjuster → Word A642
Đầu nối chiết áp Analog ngòai → Word A643



Hình 6: Sơ đồ nối đầu nối chiết áp ngòai

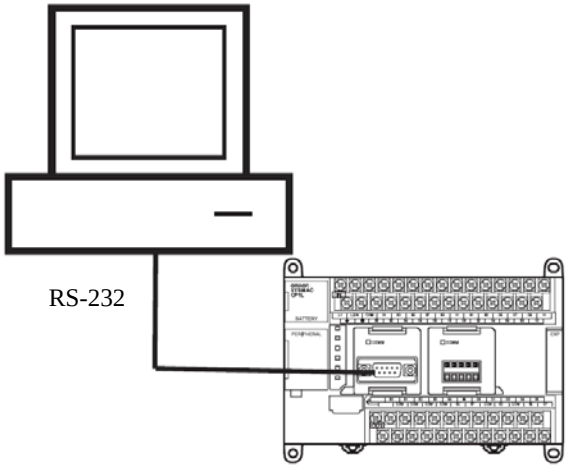


1.14 Giao tiếp truyền thông (Communications)

1.14.1) Giao tiếp dùng Host Link

Giao tiếp dùng giao thức Host Link của Omron cho phép tới 32 bộ PLC có thể được kết nối với 1 máy tính chủ (Host Computer). Host Link có thể dùng trên đường truyền RS-232C hoặc RS-422C. Khi dùng RS-232C chỉ cho phép kết nối 1:1 giữa 1 PLC với 1 computer trong khi kết nối dùng RS-422 cho phép kết nối tới 32 PLC trên mạng với 1 máy tính (1:n). Có thể dùng cổng RS-232C hoặc cổng RS-422C.

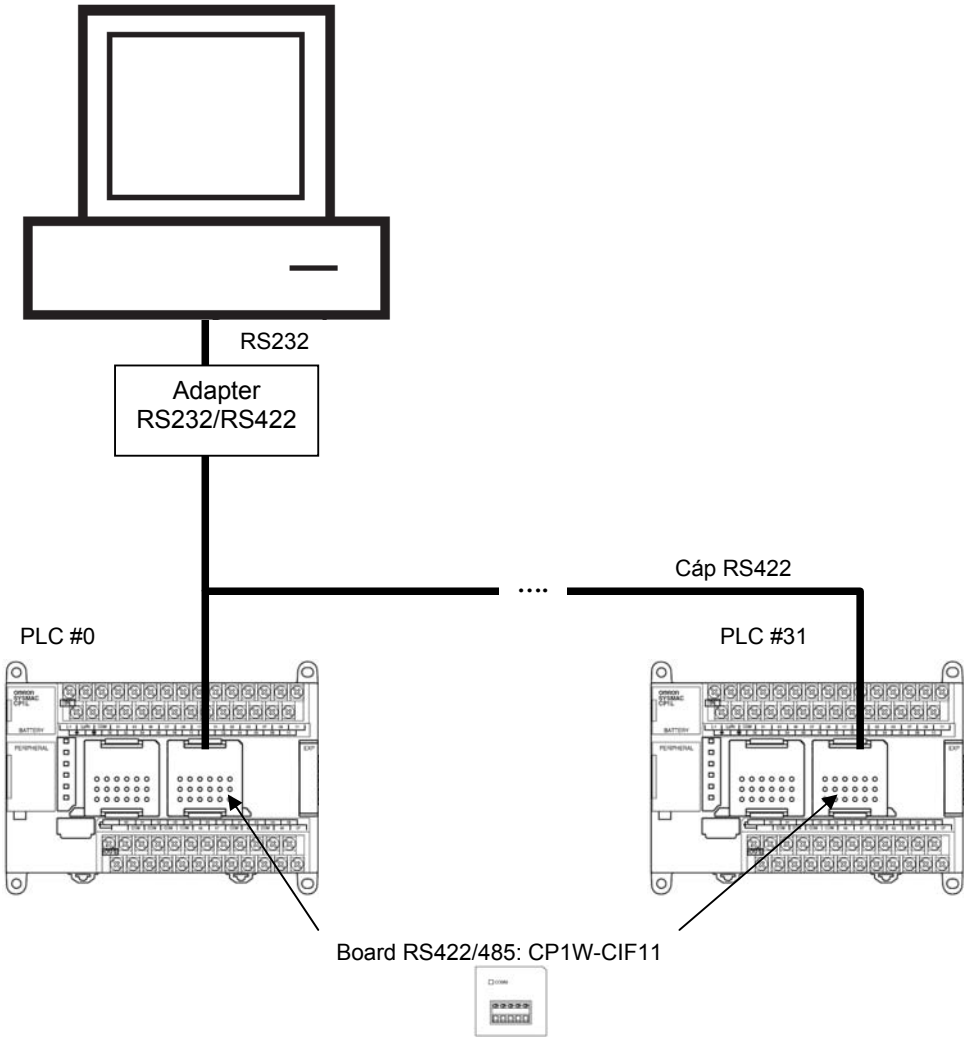
- **Kết nối 1:1**



Hình 7 : Kết nối 1:1 Host Link giữa PLC và máy tính

- **Kết nối 1:n**

Sơ đồ sau đây cho phép kết nối tới 32 PLC với 1 máy tính dùng cáp truyền RS-422.



- Khoảng cách tối đa khi dùng cáp RS-422 là 500m.

Bảng 4 Loại adapter dùng cho kết nối 1:n:

Loại	Công dụng	Mã
Link Adapter	Chuyển đổi giữa 2 chuẩn điện RS-232C và RS-422	NT-AL001 Hoặc K3SC-10

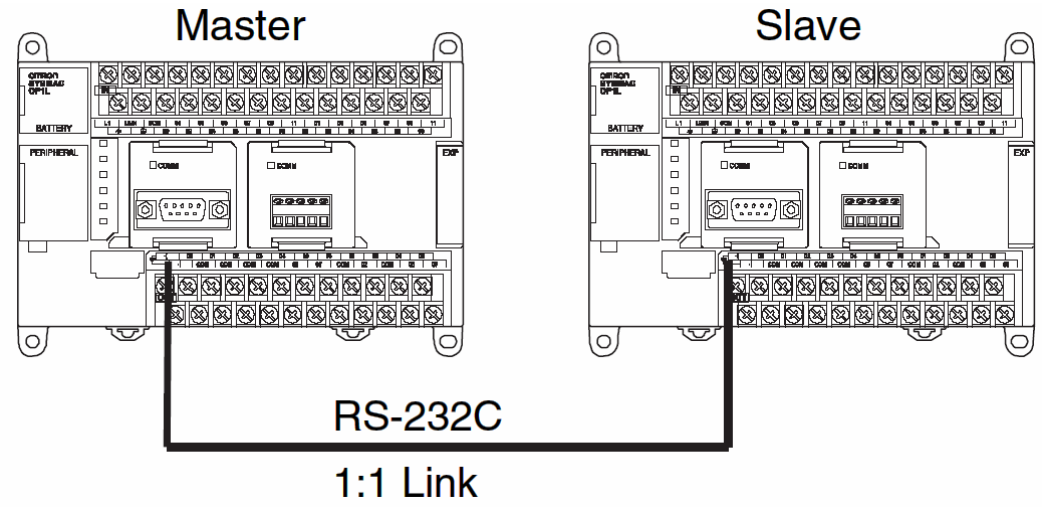
Chi tiết về các bộ lệnh Host Link cho lập trình phần mềm kết nối giữa PLC với máy tính, xin tham khảo cuốn "Programming Manual" và "Operation Manual" của CP1L/1H.

1.14.2) Liên kết dữ liệu 1: 1 giữa 2 PLC (1:1 PC Link)

Có thể thiết lập một liên kết dữ liệu (data link) của bộ nhớ giữa 1 bộ CP1L/1H với 1 bộ PLC loại CPM1/2(A), CP1L/1H, CQM1, C200HS, C200HE/G/X hay SRM1. Để thực hiện liên kết cần có cáp RS-232C. Sau khi liên kết dữ liệu giữa 2 PLC đã được tạo lập, dữ liệu trong vùng liên kết của 2 PLC này sẽ được tự động trao đổi giữa 2 PLC mà không cần lập trình.

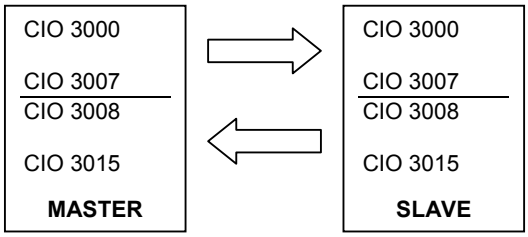
Loại	Công dụng	Model N^0
Cáp nối	Để nối giữa các PLC với nhau (chuẩn RS-232C) (max. 15m)	Tự tạo hoặc mua

Hình 9: Kết nối 1:1 PLC Link dùng cổng Peripheral Port (hình trên) và cổng RS-232C (hình dưới)



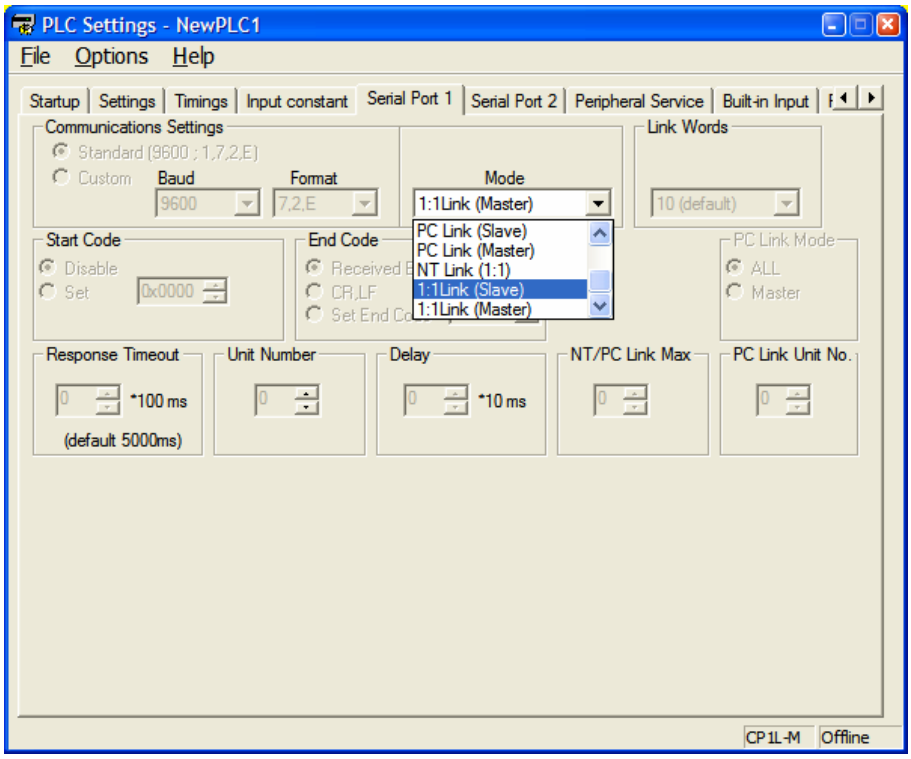
© **Ví dụ về liên kết 1:1 giữa 2 bộ CP1L/1H**

Trong mỗi bộ CP1L/1H, có một vùng bộ nhớ đặc biệt gọi là "1:1 Link Area" làm nhiệm vụ trao đổi dữ liệu giữa 2 PLC đã được thiết lập kết nối dữ liệu kiểu 1:1. Đây là các thanh ghi 16 bit có địa chỉ từ CIO 3000 đến CIO 3015 (tổng cộng 16 word/128 bit), trong đó 8 word cho việc ghi, 8 word cho việc đọc (Lưu ý: các series PLC loại C Series dùng vùng nhớ LR cho 1:1 Link Area). Khi kết nối, một PLC phải được đặt là **master**, còn PLC kia là **slave**.



Bước 1: **Đặt thông số trong PLC**

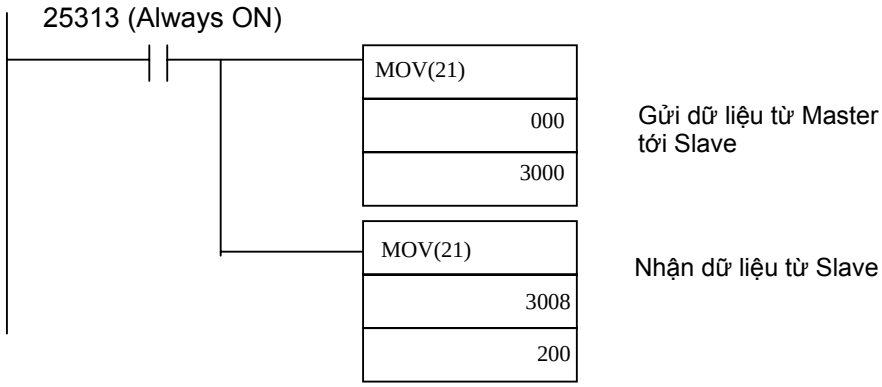
Để đặt chế độ kết nối truyền tin giữa 2 PLC, phần settings *mỗi* bộ CP1L/1H phải được đặt phù hợp, trong đó có 1 bộ là 1:1 Link Master, còn bộ kia là 1:1 Link Slave như trong hình dưới.
Việc đặt settings của từng bộ PLC được thực hiện trong tab Settings trong phần mềm CX-Programmer (xem chương 3), sau đó download vào trong PLC.



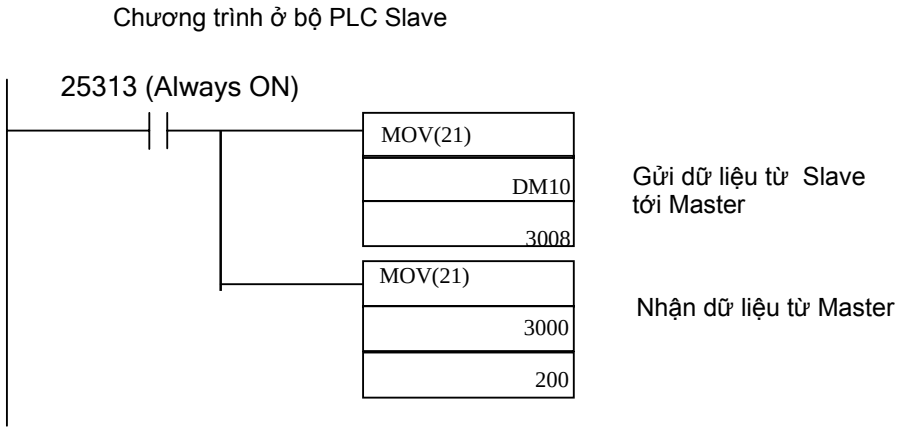
Bước 2 : **Viết chương trình truyền và nhận dữ liệu**

Mỗi bộ CP1L/1H sẽ tự động trao đổi dữ liệu với bộ PLC kia mà ta không cần lập trình. Tuy nhiên để truyền đúng dữ liệu mong muốn và nhận kết quả vào 1 bộ nhớ riêng, cần thực hiện chương trình có dạng tương tự sau đây :

Chương trình ở bộ PLC Master



Ở bộ CP1L/1H Master



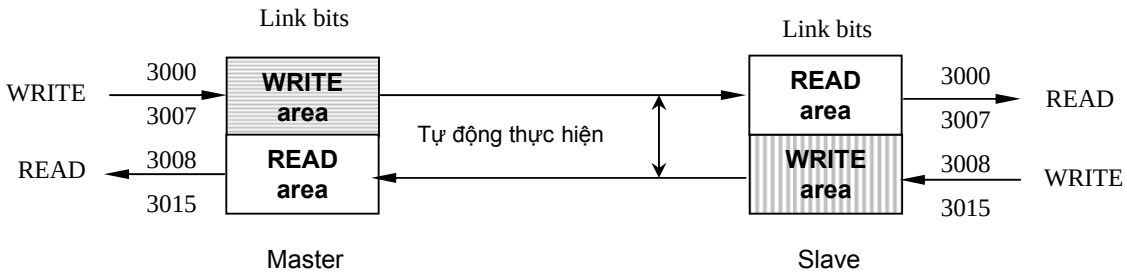
ở bộ CP1L/1H Slave

Hoạt động của hệ thống 1:1 Link

Sau khi 2 PLC chuyển sang chế độ RUN và các cáp, thông số thiết lập đã được cấu hình đúng, dữ liệu ở các vùng thanh ghi 1:1 Link Area ở 2 bộ PLC sẽ được tự động trao đổi.

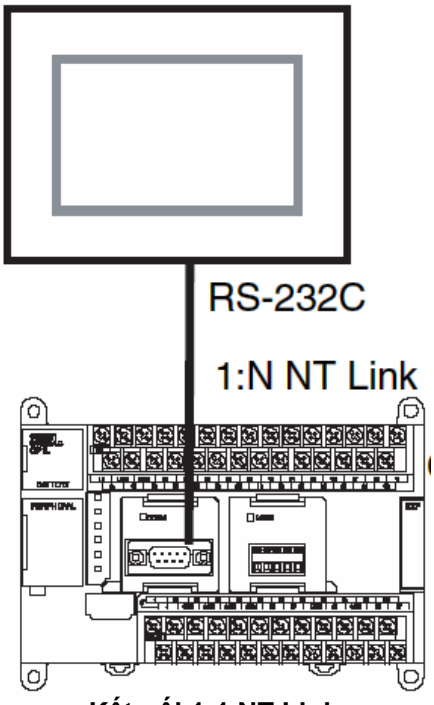
ở bộ PLC master : Dữ liệu từ thanh ghi [IR] 000 được chuyển (bằng lệnh MOV) vào thanh ghi 3000. Sau đó, dữ liệu ở 3000 của bộ Master được tự động truyền sang thanh ghi 3000 ở PLC slave đồng thời dữ liệu từ thanh ghi 3008 (nhận được từ PLC slave) được chuyển (copy) vào thanh ghi 200 của PLC master.

ở bộ PLC Slave : Dữ liệu từ thanh ghi DM10 được chuyển vào thanh ghi 3008. Sau đó, dữ liệu ở 3008 của bộ Slave được tự động truyền sang thanh ghi 3000 ở PLC Master đồng thời dữ liệu từ thanh ghi 3000 (nhận được từ PLC master) được chuyển vào thanh ghi 200 của PLC slave.



1.14.3) Truyền thông dùng NT Link

NT Link cung cấp phương tiện trao đổi dữ liệu nhanh bằng phương thức truy cập trực tiếp giữa bộ CP1L/1H với màn hình Programmable Terminal-PT trực tiếp với cổng RS-232C hoặc RS-422/485 (cần có card RS232 hoặc RS-422/485 cắm trên CP1L/1H).

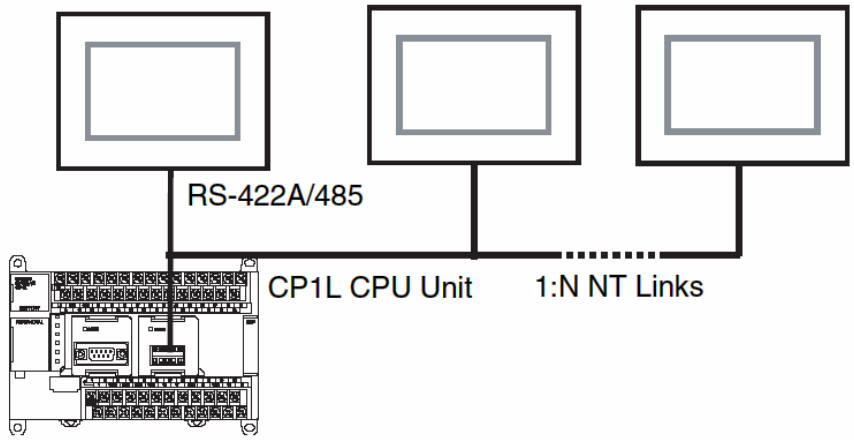


Kết nối 1:1 NT Link

Bộ	Công dụng	Model N^0
Cáp nối RS-232C	Nối giữa bộ chuyển đổi và cổng của PT	Tự tạo hoặc mua

Kết nối 1:N NT Link

Kết nối giữa cổng RS-422/485 trên CP1L với nhiều màn hình.



Bộ	Công dụng	Model N^0
Cáp nối RS-422/485	Nối giữa bộ chuyển đổi và cổng của PT	Tự tạo hoặc mua
Bộ chuyển đổi RS-422/485	Chuyển đổi từ chuẩn RS232 sang RS-422/485 cho các cổng RS232 của màn hình	

Chương 2

Các lệnh lập trình bậc thang và mnemonic

2. Bước đầu với lập trình (Programming)

2.1 Các chế đô làm việc của PLC

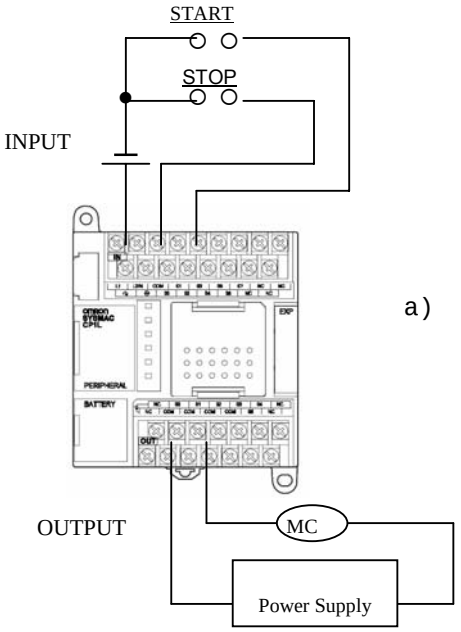
PLC có thể được đặt một trong 3 chế độ từ phần mềm lập trình CX-Programmer.

3 chế độ làm việc của PLC

- **PROGRAM** mode : Là chế độ dùng khi viết chương trình hay thực hiện các thay đổi hoặc sửa đổi đối với chương trình hiện hành
- **MONITOR** mode : Là chế độ được dùng khi thay đổi nội dung bộ nhớ trong khi PLC đang chạy (Run).
- **RUN** mode : Là chế độ dùng để thực hiện (chạy) chương trình mà ta đã lập và nạp vào PLC. Chương trình bên trong PLC không thể được thay đổi khi đang ở trong chế độ này.

Theo mặc định, PLC của Omron đều có thể được lập trình song song bằng 2 ngôn ngữ: Dòng lệnh (Statement List hay mnemonic code) & Sơ đồ bậc thang (Ladder diagram). Trong tài liệu này sẽ chủ yếu trình bày về Sơ đồ bậc thang, kèm theo bên cạnh là các lệnh tương ứng tương đương dạng Dòng lệnh (Statement List).

2.1.1) Ví dụ về một mạch tự giữ (self-holding)



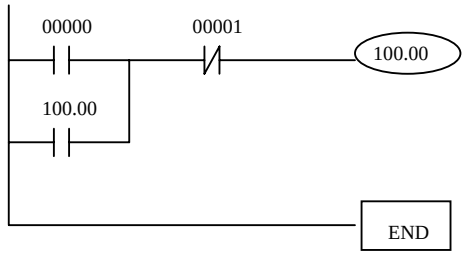
a)

Input	Thiết bị ngoài	Output	Thiết bị

00000	Nút bấm Start
00001	Nút bấm Stop

	ngoài
100.00	Motor

Ladder Diagram



b)

Mnemonic Codes

Đ. chỉ	Lệnh	Th. số
00000	LD	00000
0001	OR	100.00
0002	AND NOT	00001
0003	OUT	100.00
0004	END(01)	

c)

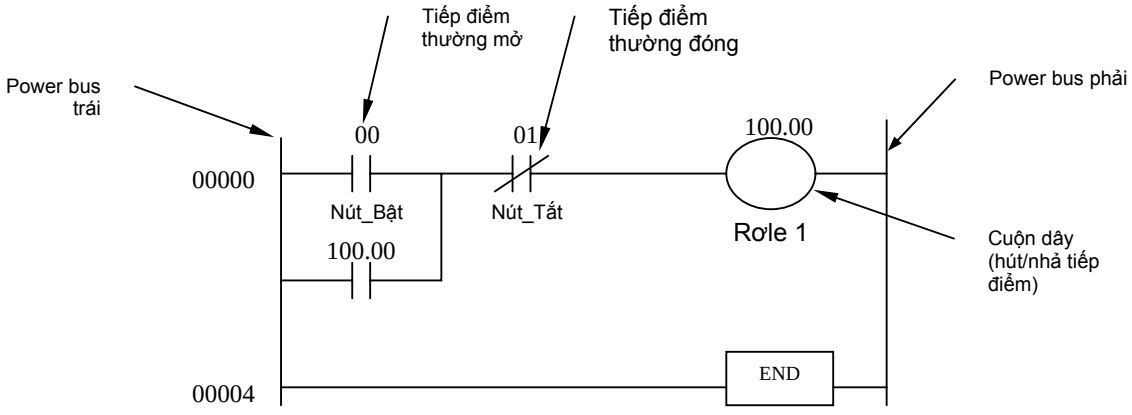
Hình 24: a) Sơ đồ nối PLC với mạch bên ngoài
b) Chương trình dạng ngôn ngữ bậc thang (Ladder Diagram)
c) Mã chương trình dạng Mnemonic Codes

Chương trình này sẽ đảm bảo đầu ra 100.00 sẽ luôn ở trạng thái ON khi 00000 lên 1 bất kể sau đó trạng thái của đầu vào 00000 như thế nào.

2.1.2) Lập trình bằng SƠ ĐỒ BẬC THANG (LADDER DIAGRAM)

Ban đầu, PLC được sử dụng chủ yếu để thay thế các sơ đồ điện phức tạp gồm rất nhiều các rơle, tiếp điểm, timer, mạch giữ, .. và các phần tử điện trung gian khác làm nhiệm vụ của các mạch logic. Tuy nhiên khi dùng PLC, các phần tử logic trung gian này được thay thế hoàn toàn bằng các sơ đồ điện "ảo" bên trong PLC do người thiết kế lập trình. Việc mô phỏng các sơ đồ điện này được lập bằng một dạng ngôn ngữ điều khiển gọi là sơ đồ bậc thang (LADDER DIAGRAM).

Ví dụ về một sơ đồ bậc thang



Thành phần cơ bản của một sơ đồ bậc thang bao gồm :

- Power bus trái và phải : giống với dây nguồn "nóng" và dây "nguội" của sơ đồ điện. Các power bus này luôn được vẽ thẳng đứng như trên hình.
- Các tiếp điểm thường đóng (NC) và thường mở (NO)
- Các cuộn dây hút/nhả các tiếp điểm khác
- Các phần tử điện khác như timer, counter,.. và các lệnh khác.

Trong sơ đồ này, cuộn dây rơle ngoài cùng bên phải sẽ chỉ nhận được điện từ power bus trái (tức dây "nóng") khi các tiếp điểm đi trước bên trái nó "cho phép" dòng điện đi qua, tức đều đóng. Do vậy các tiếp điểm (và tổ hợp đấu nối của chúng) thường được gọi là *điều kiện thực thi* (execution condition) cho cuộn dây hay các lệnh khác đi sau.
Các cuộn dây, các tiếp điểm và một số các phần tử khác luôn có một địa chỉ trong bộ nhớ để tham chiếu và sử dụng trong chương trình. Địa chỉ này được ghi phía trên ký hiệu của phần tử như trên hình. Còn các tên mô tả chức năng của chúng như Nút_Bật, Nút_Tắt, .. được ghi bên dưới. Địa chỉ của tiếp điểm sẽ điều khiển (đóng/mở) tiếp điểm này; ngược lại, cuộn dây lại điều khiển bật tắt ON/OFF địa chỉ đi kèm của cuộn dây.

2.2 Các lệnh lập trình cơ bản

PLC thường được lập trình bằng một ngôn ngữ mô phỏng giống như sơ đồ điện gọi là Ladder Diagram. Mỗi phần tử của sơ đồ là một lệnh (Instruction). Các lệnh phức tạp thường có một mã lệnh (Code) riêng.

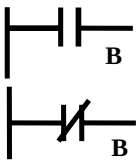
2.2.1) Lệnh tiếp điểm: Load (LD) và Load Not (LD NOT)

Lệnh LOAD hay LOAD NOT là lệnh tiếp điểm thường hở & tiếp điểm thường đóng, dùng làm điều kiện khởi đầu một thang mới trong sơ đồ bậc thang và có chức năng giống với một tiếp điểm của sơ đồ điện. Các tiếp điểm khi nối với các phần tử khác thường đóng vai trò làm **điều kiện thực hiện** (execution condition) cho các phần tử đi sau nó. Lệnh này luôn được gán với một địa chỉ bit xác định trạng thái của tiếp điểm này.
Chú ý là 2 lệnh này luôn luôn nằm ở phía trái nhất của một khối logic trong sơ đồ bậc thang (nghĩa là không có một lệnh nào loại khác được phép nằm ở phía trái của lệnh này trong khối logic).
Có 2 loại:

- Lệnh LD : Tương đương với một tiếp điểm thường mở (Normally Open - NO) trong sơ đồ điện. Khi bit đi kèm là 1 (ON), tiếp điểm sẽ đóng và các phần tử (lệnh) đi sau tiếp điểm sẽ được hoạt động (có điện) và ngược lại khi bit đi kèm là 0 (OFF), tiếp điểm sẽ mở và các phần tử đi sau tiếp điểm sẽ không được hoạt động (không có điện chạy qua tiếp điểm)
- Lệnh LD NOT : Tương đương với một tiếp điểm thường đóng (Normally Closed - NC) trong sơ đồ điện. Khi bit đi kèm là 0 (OFF), tiếp điểm sẽ đóng và các phần tử (lệnh) đi sau tiếp điểm sẽ được hoạt động (có điện) và ngược lại khi bit đi kèm là 1 (ON), tiếp điểm sẽ mở và các phần tử đi sau tiếp điểm sẽ không được hoạt động (không có điện chạy qua tiếp điểm)

LOAD-**LD (Normally open) →**

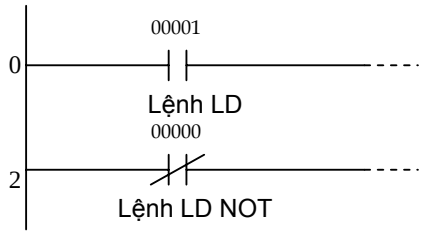
LOAD NOT-**LD NOT (Normally Closed) →**



B : BIT
IR, SR, AR, HR, TC, LR,

Các địa chỉ có thể truy cập ở dạng bit

Ví dụ :



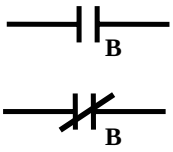
Địa chỉ	Lệnh	Th. số
00000	LD	00000
00001	Lệnh khác...	.....
00002	LD NOT	00000
00003	Lệnh khác	.....

2.2.2) Lệnh tiếp điểm: AND và AND NOT

Lệnh AND (AND NOT) dùng để tạo ra các tiếp điểm thường mở (thường đóng) theo sau (nối tiếp) với các tiếp điểm tạo ra bởi lệnh LD hay LD NOT.

AND-**AND**

AND NOT-**AND NOT**



B : BIT
IR, SR, AR, HR, TC, LR

Các địa chỉ có thể truy cập ở dạng bit

Ví dụ: AND, AND NOT

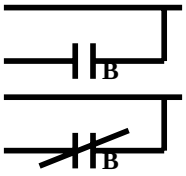
Địa chỉ	Lệnh	Th. Số
00000	LD	00000
00001	AND NOT	00100
00002	AND	LR 00000
00003	Lệnh ..	

2.2.3) Lệnh tiếp điểm: OR, OR NOT

Lệnh OR (OR NOT) tạo ra các tiếp điểm thường mở (thường đóng) nối song song với một nhánh khác.

OR-**OR**

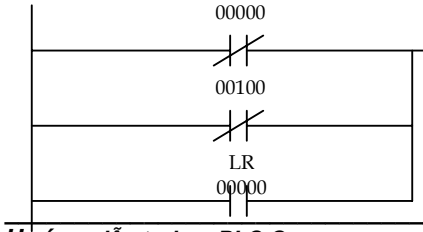
OR NOT-**OR NOT**



B : BIT
IR, SR, AR, HR, TC, LR

Các địa chỉ có thể truy cập ở dạng bit

Ví dụ : OR, OR NOT



Instruction.

Địa chỉ	Lệnh	Tham số

00000	LD NOT	00000
00001	OR NOT	00100
00002	OR	LR 00000
00003	Instruction	

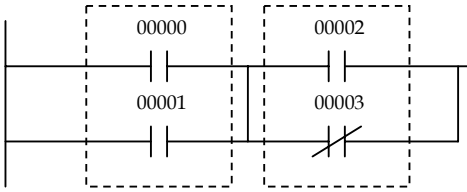
2.2.4) Lệnh AND LD và OR LD

AND LOAD-(**AND LD)** và OR LOAD-(**OR LD)**

- Lệnh AND LD nối tiếp 2 khối logic với nhau trong một sơ đồ bậc thang.
- Lệnh OR LD nối song song 2 khối với nhau trong một sơ đồ bậc thang

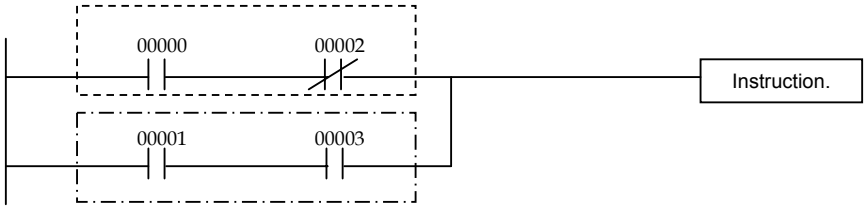
Cần chú ý thứ tự nhập lệnh này: các khối logic cần nối với nhau được nhập riêng rẽ trước, sau đó mới nhập lệnh OR LD hoặc AND LD.
Lệnh này không cần tham số & chỉ cần dùng khi viết chương trinh dạng mnemonic.

Ví dụ: AND LD



Đ. chỉ	Lệnh	Th. số
00000	LD	00000
00001	OR	00001
00002	LD	00002
00003	OR NOT	00003
00004	AND LD	
. .	. .	. .

Ví dụ OR LD

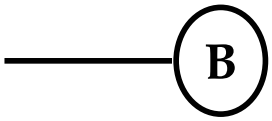


Địa chỉ	Lệnh	Th. số
00000	LD	00000
00001	AND NOT	00001
00002	LD	00002
00003	AND	00003
00004	OR LD	
00005	Lệnh ...	.....

2.2.5) Lệnh cuộn dây: OUT và OUT NOT

Lệnh OUT (OUT NOT) sẽ bật bit được gán cho lệnh này lên ON (xuống OFF) khi điều kiện thực thi đi trước nó là ON và sẽ reset bit này về OFF khi điều kiện đi trước là OFF. Lệnh OUTPUT giống với chức năng ***cuộn dây*** trong sơ đồ điện là khi một ***cuộn dây*** nhận được điện từ tiếp điểm (điều kiện) đi trước nó sẽ hút (đóng) hay nhả (mở) tiếp điểm đi kèm.

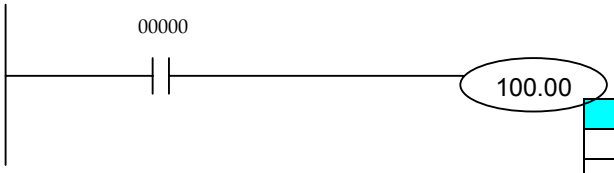
Ký hiệu: OUTPUT-**OUT**



B : BIT
IR, SR, AR, HR, LR, TR

Các địa chỉ có thể truy cập ở dạng bit

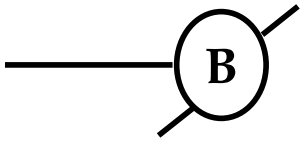
Ví dụ: Lệnh OUT



Đ. chỉ	Lệnh	Th. số
00000	LD	00000
00001	OUT	100.00

Tiếp điểm 00000 là điều kiện thực thi của cuộn dây 100.00.

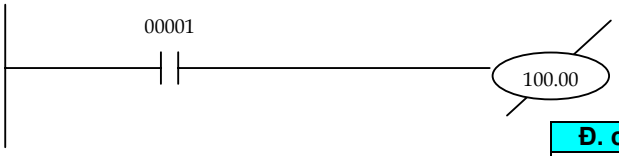
Ký hiệu: OUTPUT NOT-**OUT NOT**



B : BIT
IR, SR, AR, HR, LR, TR

Các địa chỉ có thể truy cập ở dạng bit

Ví dụ: **OUT NOT**



Đ. chỉ	Lệnh	Th. số
00000	LD	00001
00001	OUT NOT	100.00

2.3 Các hàm chức năng đặc biệt - Function (FUN)

Ngoài các lệnh điều kiện và đầu ra đơn giản trên, trong PLC loại CP1L/1H còn có các lệnh với các chức năng phức tạp khác. Mỗi lệnh này đều có một mã lệnh (code) riêng. Khi dùng CX-Programmer, ta sẽ dùng công cụ **Instruction** để thêm 1 hàm chức năng và có thể nhập mã lệnh hoặc tên lệnh đều được.
Dưới đây là mã của một số lệnh trong PLC loại CP1L/1H :

FUN 01	là lệnh	END (End Instruction)
FUN 02	,,	IL (Interlock)
FUN 03	,,	ILC (Interlock Clear)
FUN 04	,,	JMP (Jump End)
FUN 05	,,	JME (Jump End)
FUN 10	,,	SFT (Shift Register)
FUN 11	,,	KEEP (Latching Relay)
FUN 12	,,	CNTR (Reversible Counter)
FUN 13	,,	DIFU (Differentation - Up)
FUN 14	,,	DIFD (Differentation -Down)

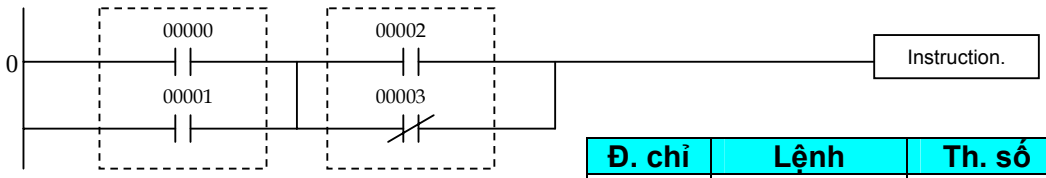
Chú ý :

- Các số 0 ở đầu các mã lệnh (ví dụ **0**1 (END), **0**2 (IL),...) phải được nhập vào. Nếu chỉ nhập chữ số sau thì kết quả có thể không đúng.
- Khi biểu diễn lệnh, người ta thường ghi kèm cả mã lệnh của lệnh đó trong dấu ngoặc đơn theo sau tên lệnh. Ví dụ: END(01), IL(02),.. Tuy nhiên khi nhập lệnh vào chương trình thì chỉ cần nhập tên lệnh hoặc mã lệnh là đủ.

2.3.1) Lệnh END (01)

Lệnh END(01) dùng để đánh dấu điểm kết thúc của chương trình. Một chương trình có thể có nhiều lệnh END (01) nhưng PLC sẽ chỉ xử lý các lệnh từ đầu chương trình đến lệnh END đầu tiên mà nó gặp, sau đó chương trình lại bắt đầu từ lệnh đầu tiên của chương trình. Nếu không có lệnh END trong chương trình, khi PLC chuyển sang chế độ RUN thì trên màn hình của bộ lập trình cầm tay sẽ báo lỗi "NO END INSTR" và chương trình sẽ không được thực hiện.

Ví dụ Chương trình dạng sơ đồ bậc thang (trên) và dạng Mnemonic tương đương (dưới) đều không có lệnh END(01), do đó sẽ bị báo lỗi và không thể chạy được:



Đ. chỉ	Lệnh	Th. số
00000	LD	00000
00001	OR	00001
00002	LD	00002
00003	OR NOT	00003
00004	AND LD	---

Lỗi sẽ báo như sau trên phần mềm CX-Programmer:

NO END INSTRUCTION!

2.3.2) Lệnh IL (02) và ILC (03)

Lệnh IL (Interlock) và ILC (Interlock Clear) luôn được dùng đi kèm với nhau. Khi một lệnh IL được đặt trước một đoạn chương trình, điều kiện thực hiện của IL sẽ điều khiển điều kiện thực hiện của toàn bộ các lệnh bắt đầu từ sau lệnh IL cho đến lệnh ILC đầu tiên sau lệnh IL này. Khi điều kiện thực hiện của lệnh IL là ON, chương trình vẫn được thực hiện bình thường. Khi điều kiện thực hiện của lệnh IL là OFF, tất cả các lệnh theo sau lệnh IL cho đến lệnh ILC đầu tiên đều được thi hành với điều kiện thực hiện là OFF. Nghĩa là các lệnh Output nằm giữa IL và ILC sẽ là OFF.

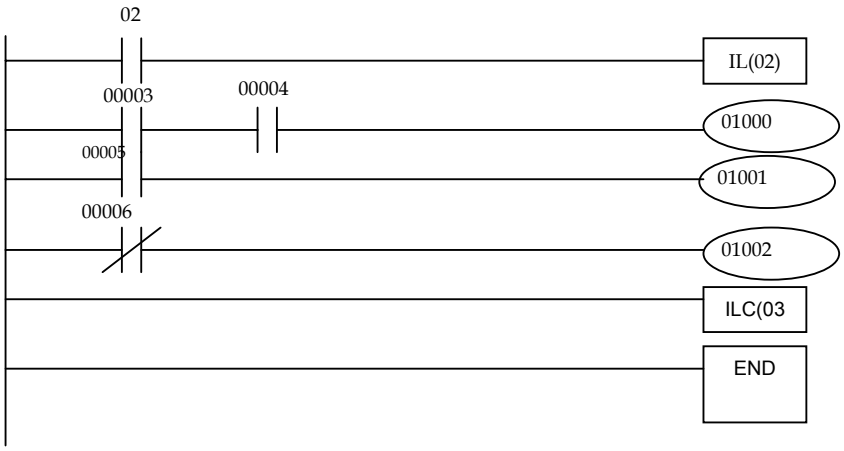
Chương trình sẽ trở lại hoạt động bình thường sau lệnh ILC.

Ví dụ:

Đ. chỉ	Lệnh	Th. số
00000	LD	00002
00001	IL (02)	-
00002	LD	00003
00003	AND	00004
00004	OUT	100.00
00005	LD	00005
00006	OUT	100.01
00007	LD NOT	00006
00008	OUT	100.02
00009	ILC(03)	-
00010	END(01)	-

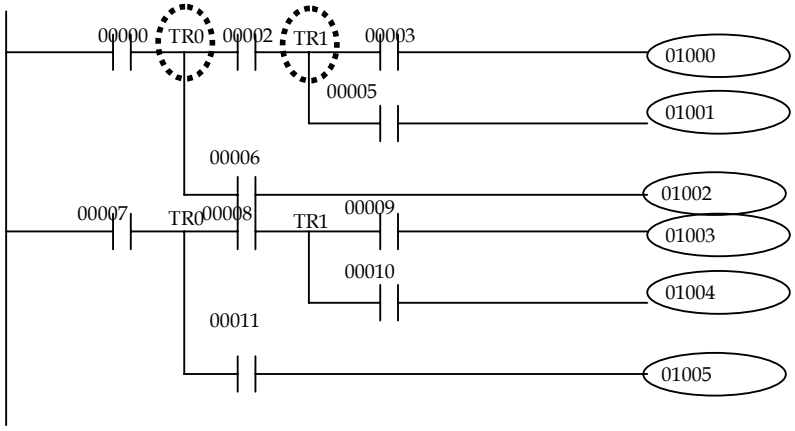
Chú ý :

- Các bit được set hoặc reset bởi lệnh KEEP đặt trong khối INTERLOCK vẫn ở trạng thái cũ của chúng.
- Timer nằm trong khối INTERLOCK sẽ bị reset khi điều kiện thực thi của IL là OFF hoặc khi mất điện.
- PV của counter nằm trong khối INTERLOCK sẽ <u>không</u> bị reset khi điều kiện thực thi của IL là OFF.

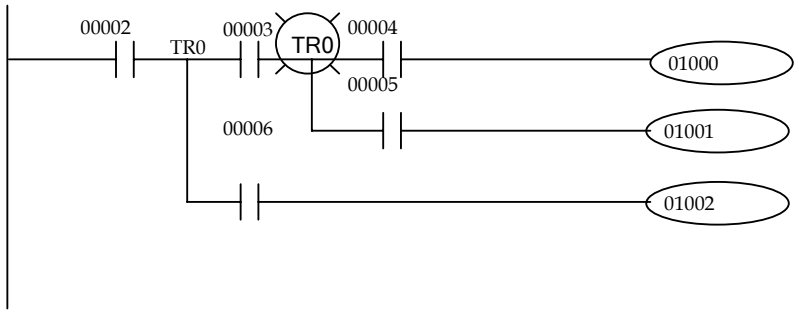


2.3.3) Bit phân nhánh - TR (Temporary Relay)

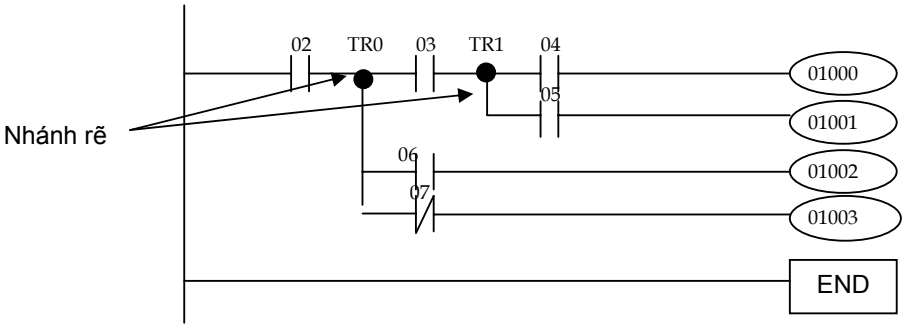
Trong các nhánh chương trình, các bit phân nhánh (7 bit từ TR0-TR7) được dùng để lưu điều kiện thực hiện tại điểm phân nhánh, giúp cho việc thực hiện chương trình tại nhánh chương trình được đúng đắn.



Chương trình sau *sai* do dùng hai lần bit TR0 trong cùng một thang chương trình:



Ví dụ : Dùng bit TR để lưu các điều kiện thực hiện khi phân nhánh



Dưới đây là chương trình trên dạng Mnemonic. Các bit TR được nhập vào bằng lệnh OUT, với tham số là số của bit TR, sau đó được dùng lại bằng lệnh LD để bắt đầu một nhánh khác của chương trình:

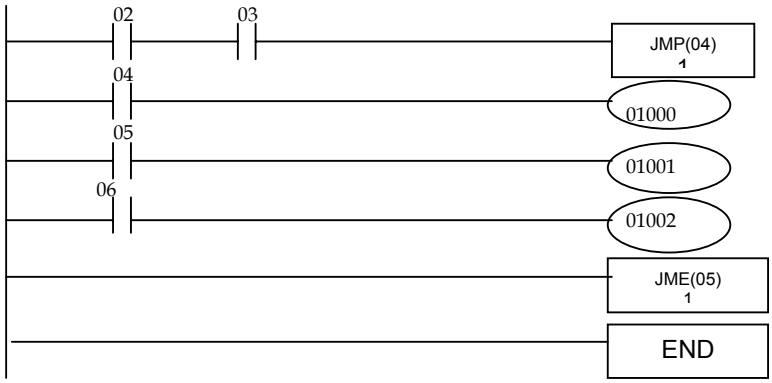
Địa chỉ	Lệnh	Th. số
0000	LD	00002
0001	OUT	TR 0
0002	AND	00003
0003	OUT	TR1
0004	AND	00004
0005	OUT	100.00
0006	LD	TR1
0007	AND	00005
0008	OUT	100.01
0009	LD	TR0
0010	AND	00006
0011	OUT	100.02
0012	LD	TR0
0013	AND-NOT	00007
0014	OUT	100.03
0015	END (01)	-

Chú ý : Các bit TR chỉ được dùng khi lập trình dạng mnemonic code. Còn khi lập trình với ladder diagram (dùng phần mềm CX-PROGRAMMER), bit này không cần thiết vì chương trình đã tự động thực hiện việc này.

2.3.4) Lệnh JMP (04) và JME (05)

Mỗi lệnh JUMP gồm cặp lệnh JMP và JME có số từ 00 đến 99; JMP và JME luôn đi theo cặp với nhau. Khi chương trình gặp lệnh JMP n (n= số của lệnh JUMP), nó sẽ bỏ qua không thực hiện các lệnh theo sau lệnh này cho đến lệnh JME n có cùng số. Khi gặp lệnh JME, chương trình sau đó lại thực hiện bình thường. Mặc dù hoạt động của JMP khá giống với hoạt động của INTERLOCK khi điều kiện thực hiện của IL là OFF, nhưng đối với lệnh JMP, các toán tử nằm giữa lệnh JMP và JME không bị OFF mà vẫn giữ nguyên trạng thái trước khi thực hiện lệnh JUMP này.

Ví du : Lệnh JUMP



Chương trình dạng mnemonic :

Đ. chỉ	Lệnh	Th. số
0000	LD	00002
0001	AND	00003
0002	JMP(04)	1
0003	LD	00004
0004	OUT	100.00
0005	LD	00005
0006	OUT	100.01
0007	LD	00006
0008	OUT	100.02
0009	JME(05)	1
0010	END(01)	

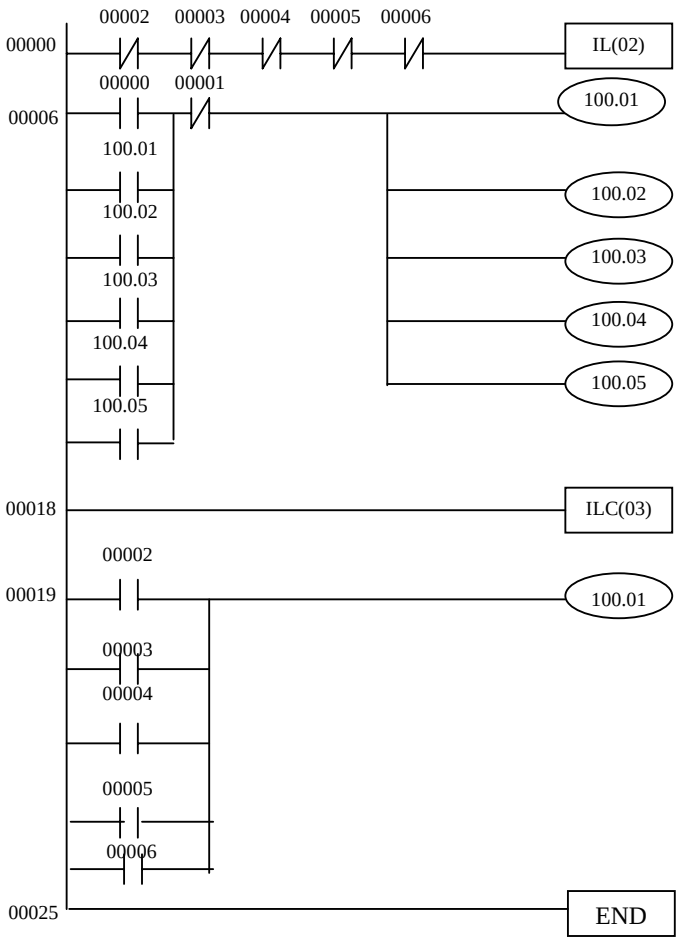
2.4 Ví dụ ứng dụng: Dừng động cơ khi có quá tải

Có 5 motor nối liên động với nhau. Khi nút PB Start được nhấn, cả 5 Motor đều khởi động và chạy nếu như không có motor nào đang bị quá tải (overload). Nếu 1 trong 5 motor này bị quá tải hoặc khi nút Stop được nhấn, cả 5 motor sẽ dừng. Đèn báo Overload sẽ sáng nếu có motor nào đó đang bị quá tải.

I/O

Đầu vào		Đầu ra	
00000	PB Start	100.00	Lamp Overload
00001	PB Stop	100.01	Motor 1
00002	Overload M1	100.02	Motor 2
00003	Overload M2	100.03	Motor 3
00004	Overload M3	100.04	Motor 4
00005	Overload M4	100.05	Motor 5
00006	Overload M5		

Chương trình dạng sơ đồ bậc thang



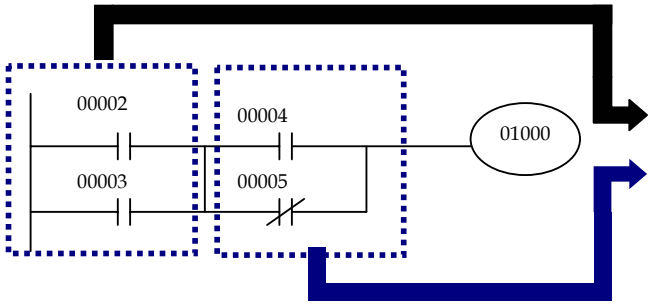
Chương trình dạng mnemonic

00000	LD	NOT	00002
00001	AND	NOT	00003
00002	AND	NOT	00004
00003	AND	NOT	00005
00004	AND	NOT	00006
00005	IL	(02)	
00006	LD		00000
00007	OR		100.01
00008	OR		100.02
00009	OR		100.03
00010	OR		100.04
00011	OR		100.05
00012	AND	NOT	00001
00013	OUT		100.01
00014	OUT		100.02
00015	OUT		100.03
00016	OUT		100.04
00017	OUT		100.05
00018	ILC	(03)	
00019	LD		00002
00020	OR		00003
00021	OR		00004
00022	OR		00005
00023	OR		00006
00024	OUT		100.01
00025	END	(01)	

- *Các lệnh **AND LD** và **OR LD** có thể được dùng để lập các sơ đồ với các phần tử kết nối phức tạp khi viết chương trình bằng mnemonic:*

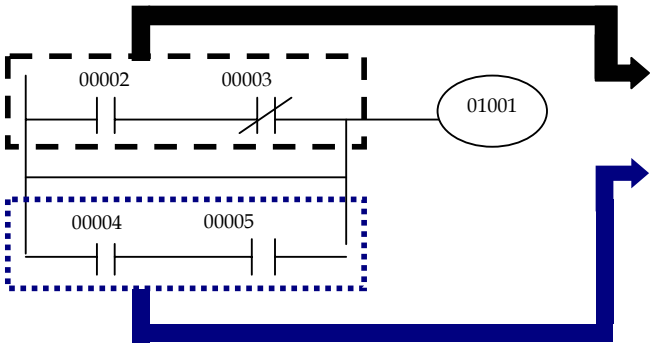
Ví dụ : Cách nhập các lệnh AND LD và OR LD:

■ **AND LD** Dùng để nối nối tiếp 2 khối logic chương trình



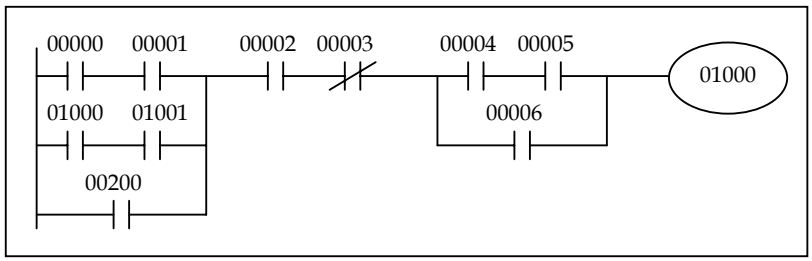
Đ. chỉ	Lệnh	Th. số
00000	LD	00002
00001	OR	00003
00002	LD	00004
00003	OR NOT	00005
00004	AND LD	- - -
00005	OUT	100.00

■ **OR LD** Dùng để nối song song 2 khối logic chương trình



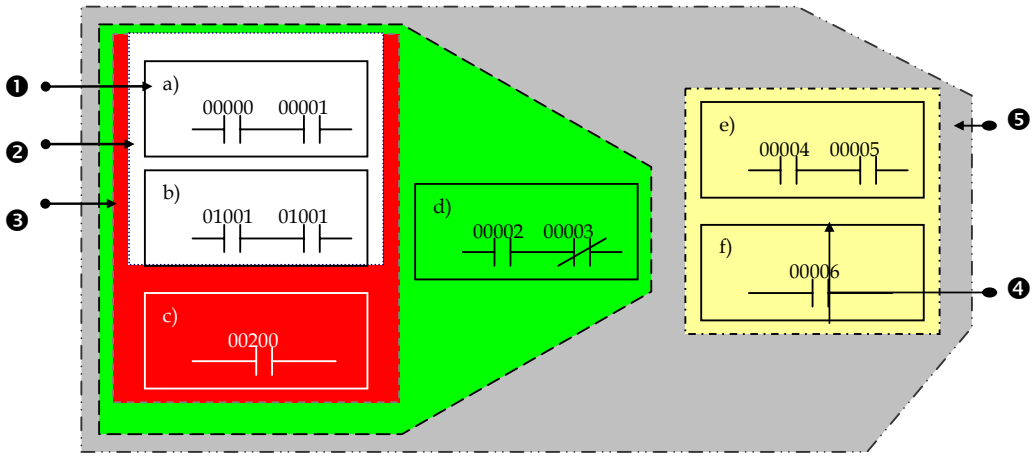
Đ. chỉ	Lệnh	Th. số
00000	LD	00002
00001	AND NOT	00003
00002	LD	00004
00003	AND	00005
0004	OR LD	- - -
00005	OUT	100.01

Ví dụ : ta có 1 đoạn chương trình với các khối logic chương trình nối kết khá phức tạp như hình dưới :

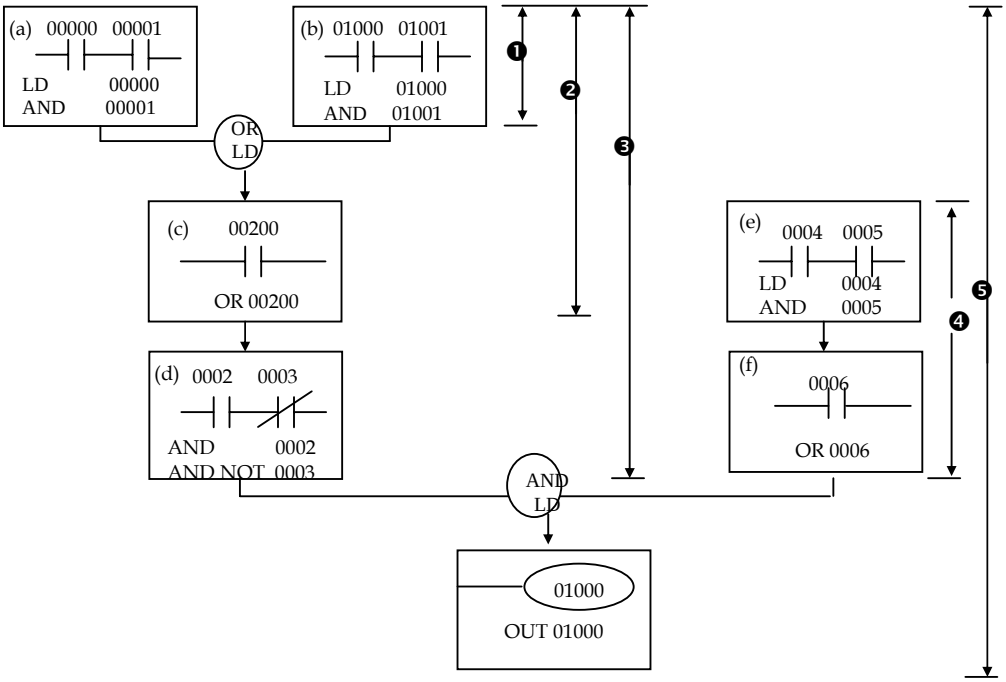


Để có thể nhập được chương trình này cũng như các khối chương trình phức tạp khác vào bằng bộ lập trình cầm tay, cần thực hiện các bước sau :

(1) Chia nhỏ đoạn chương trình thành các khối **block cơ bản [1] - [5]**



2) Nhập từng khối này vào bộ lập trình, bắt đầu từ trên xuống dưới, từ trái qua phải như bình thường theo thứ tự các khối ❶ → ❺ trên ví dụ dưới đây:

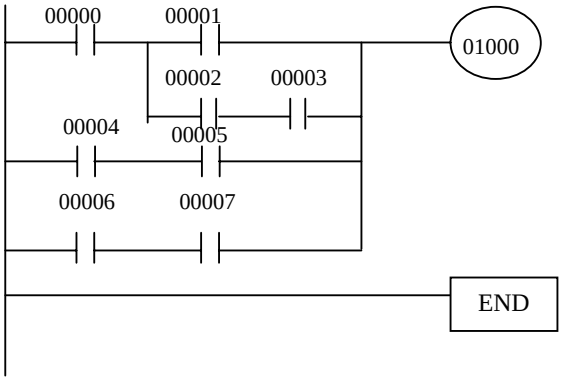


Chú ý : Các khối logic cơ bản là các khối với các phần tử có thể được nối với nhau bằng các lệnh LD, LD NOT, AND, AND NOT, OR, OR NOT, ..

Bài ôn tập :

Cho một chương trình dưới dạng Ladder Diagram dưới đây. Hãy nhập vào chuyển đổi chương trình sang dạng Mnemonic Code :

Ladder Diagram

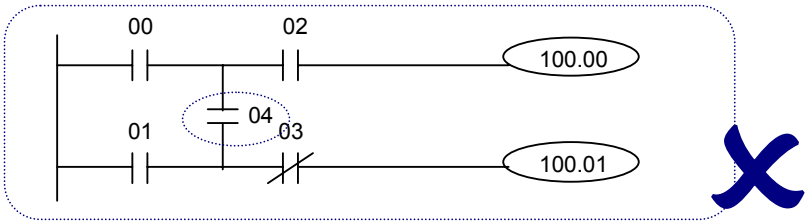


Mnemonic Codes

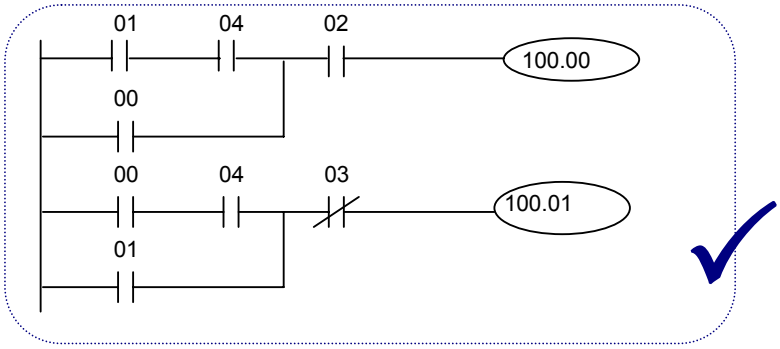
Đ. chỉ	*Lệnh*	*Th.số*

2.5 Các lưu ý khi lập một chương trình dạng Ladder Diagram

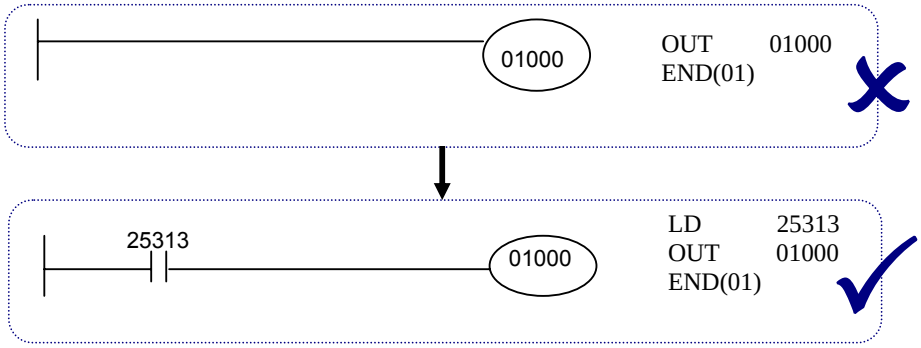
1. Hai thang khác nhau không được phép nối bằng một tiếp điểm thẳng đứng :



Đoạn chương trình trên không đúng vì hai thang được nối với nhau bằng một tiếp điểm thẳng đứng và sẽ được sửa như đoạn chương trình dưới đây :

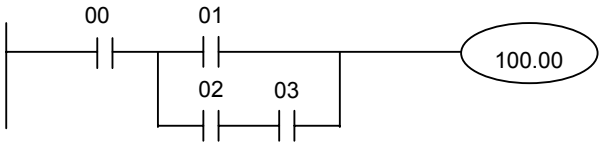


2. Nếu một lệnh OUTPUT hoặc một FUN luôn luôn cần điều kiện thực hiện là ON, lệnh này không được nối trực tíếp với thanh power bus bên trái. Thay vào đó, phải nối qua một tiếp điểm dùng cờ "ALWAYS ON" (có địa chỉ là 25313)



Trường hợp ngoại lệ : Các lệnh INTERLOCK CLEAR, JUMP END, STEP, END không tuân theo quy tắc này.

3. Chú ý đến các số lượng lệnh cần thiết để nhập một chương trình.



Hình A :

Ở sơ đồ hình A trên, ta cần có thêm lệnh OR LD và AND LD để nối nhánh dưới với nhánh trên.

Các lệnh dạng mnemonic cho sơ đồ hình A

Đ. chỉ	*Lệnh*	*Th. số*
0000	LD	00
0001	LD	01
0002	LD	02
0003	AND	03
0004	OR LD	
0005	AND LD	
0006	OUT	100.00

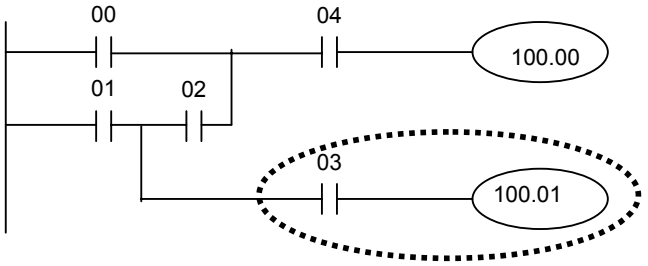
Đoạn chương trình trên có thể được sửa lại như hình B sau đây :

Các lệnh dạng mnemonic cho sơ đồ hình B

Đ. chỉ	*Lệnh*	*Th. số*
0000	LD	02
0001	AND	03
0002	OR	01
0003	AND	00
0004	OUT	100.00

Rõ ràng là với cách biểu diễn tương đương như hình B, việc biểu diễn đơn giản hơn và giảm đi được 2 lệnh AND LD và OR LD.

4. Một nhánh không được xuất phát từ một nhánh song song khác.

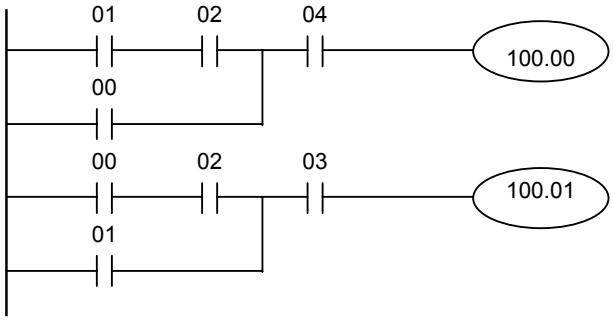


Khi các bit 01, 02, 04 là ON sẽ bật 1000 lên ON

Hình A

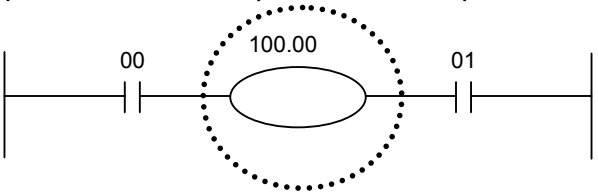
Ở hình A, ta muốn khi các bit 00, 02 và 03 là ON hoặc khi 01 và 03 là ON, bit 100.01 sẽ được bật lên ON. Tuy nhiên đó là cách biểu diễn không thích hợp với việc nhập bằng Console.

Đoạn chương trình trên được sửa lại như hình B sau :

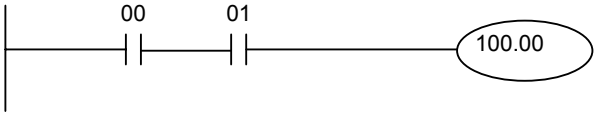


Hình B

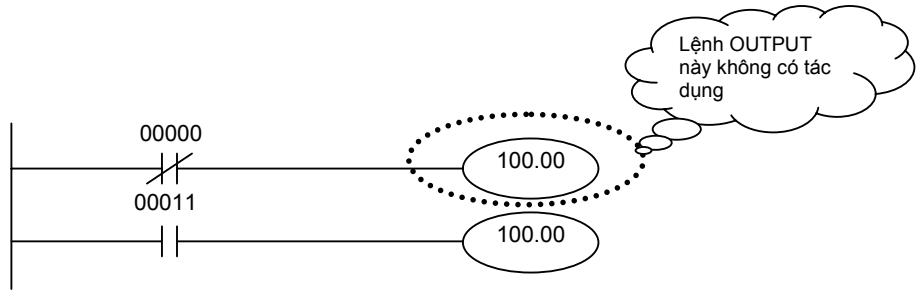
5. Lệnh OUT hoặc OUT NOT (nếu có) phải là lệnh cuối cùng trên thang và phải được nối vào power bus bên phải.



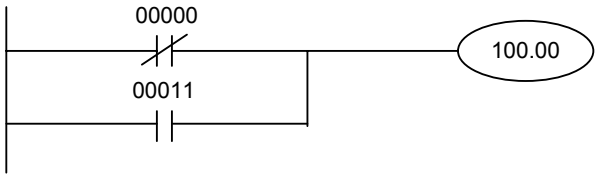
Đoạn chương trình trên không đúng vì lệnh OUT 100.00 không được nối trực tiếp vào power bus mà qua một tiếp điểm và sẽ được sửa lại như sau :



6. Nếu một địa chỉ bit được dùng lặp lại trên hai lệnh OUTPUT khác nhau, lệnh OUTPUT đi trước sẽ không có tác dụng.



Do đó, nếu 2 bit 00000 và 00011 đều dùng để điều khiển lệnh OUTPUT với bit 100.00 thì đoạn chương trình trên sẽ được sửa lại như sau :



Bài ôn tập về Lập trình cơ bản trên CP1L/1H dùng Mnemonic Code và Ladder Diagram

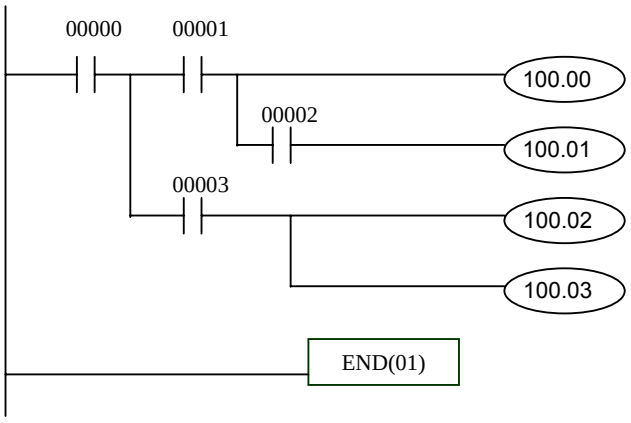
1. Hãy viết chương trình dạng Mnemonic Code cho chương trình dạng sơ đồ bậc thang dưới đây

Đ. chỉ	*Lệnh*	*Th. số*
00000		
00001		
00002		
00003		
00004		
00005		
00006		
00007		
00008		
00009		

2. Cho một chương trình dạng Mnemonic Code bên dưới, hãy viết chương trình tương đương dưới dạng Ladder Diagram :

Đ. chỉ	*Lệnh*	*Th. số*
00000	LD	00000
00001	AND	00001
00002	LD NOT	00002
00003	AND	00003
00004	OR LD	
00005	LD	00004
00006	AND NOT	00005
00007	LD NOT	00006
00008	AND	00007
00009	OR LD	
00010	AND LD	
00011	OUT	100.00
00012	END (01)	

3. Hãy nhập chương trình dưới dạng Mnemonic Code cho đoạn chương trình dạng sơ đồ bậc thang dưới đây :



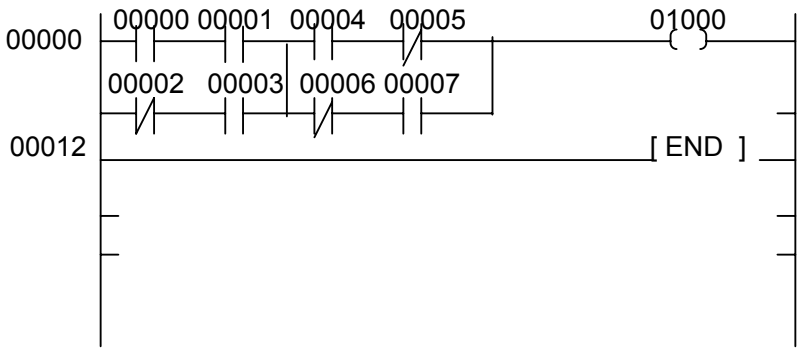
Đ. chỉ	*Lệnh*	*Th. số*
00000		
00001		
00002		
00003		
00004		
00005		
00006		
00007		
00008		
00009		
00010		

Đáp án :

1)

```
00000  LD          00000
00001  AND         00001
00002  AND NOT     00002
00003  LD          00003
00004  AND NOT     00004
00005  OR   LD
00006  AND         00005
00007  AND         00006
00008  OUT         100.00
00009  END(01)
```

2)



3)

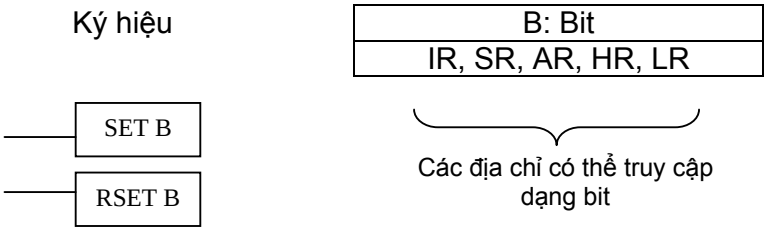
```
00000  LD              00000
00001  OUT       TR      0
00002  AND             00001
00003  OUT             100.00
00004  AND             00002
00005  OUT             100.01
00006  LD        TR      0
00007  AND             00003
00008  OUT             100.02
00009  OUT             100.03
00010  END(01)
```

Chú ý : Nhánh rẽ với lệnh AND 00002 và OUT 00001 không cần thêm bit TR vì giữa điểm rẽ nhánh và lệnh OUT 100.00 không có tiếp điểm nào.

2.6 Các lệnh đặc biệt thông dụng khác:

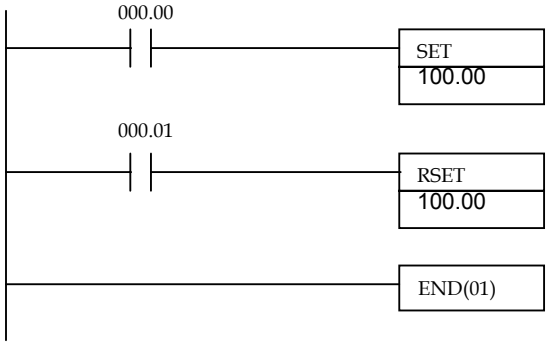
2.6.1) Bật bit (SET) và Xoá bit (RESET) SET- RSET

Lệnh SET sẽ bật bit đi kèm lên ON khi điều kiện thực thi của nó là ON. Sau đó, bit sẽ vẫn ở trạng thái ON không phụ thuộc vào việc lệnh SET có điều kiện thực hiện là ON hay OFF cho đến khi lệnh RESET (RSET) xoá nó về OFF.



Ví dụ: Bit 100.00 sẽ được bật lên ON khi điều kiện thực hiện của lệnh SET (là bit 00000) là ON. ở các chu kỳ quét sau, bit 100.00 sẽ vẫn giữ (Hold) ở trạng thái ON cho dù bit 00000 là ON hay OFF. Bit 100.00 sẽ chỉ bị xoá bởi lệnh Reset khi bit 00001 là ON.

Sơ đồ bậc thang



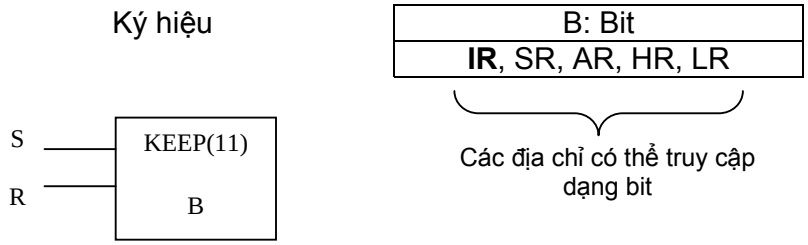
Các lệnh dạng Mnemonic

```
00000   LD      000.00
00001   SET     100.00
00002   LD      000.01
00003   RSET    100.00
00004   END
```

Chú ý : Trạng thái của bit được SET hay RSET sẽ không thay đổi khi nằm trong khối INTERLOCK hay JUMP.

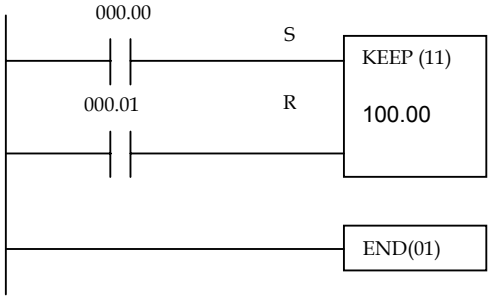
2.6.2) Lệnh giữ KEEP - KEEP(11)

Lệnh KEEP hoạt động như một rơ le chốt với hai đầu vào là SET (S) và RESET (R). Bit B sẽ được Set lên ON khi đầu vào S là ON và sẽ vẫn giữ ở ON cho đến khi B bị reset về OFF khi đầu vào R là ON.



Chú ý : Các bit được set hay reset bởi KEEP không bị reset khi nằm trong khối INTERLOCK.

Ví dụ : Bit 100.00 sẽ được set lên ON khi bit 00000 lên ON và sẽ vẫn ở ON cho dù sau đó bit 00000 là ON hay OFF. Bit 100.00 chỉ bị reset về OFF khi bit 00001 là ON (đầu vào RESET sẽ tác động)

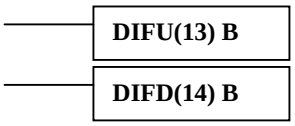


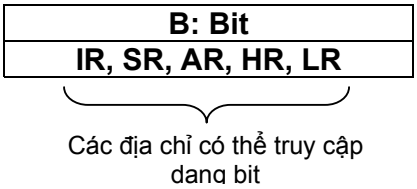

```
00000   LD      000.00
00001   LD      000.01
00002   KEEP    100.00
00003   END
```

2.6.3) DIFFERENTIATE UP và DOWN - DIFU(13) & DIFD(14)

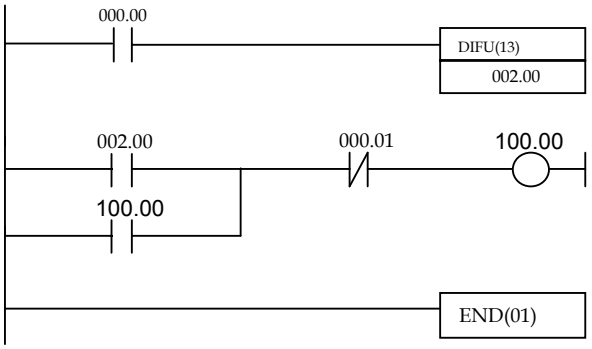
- **DIFU(13)** : Lệnh này sẽ bật bit đi kèm lên 1 trong vòng một chu kỳ quét (scan/cycle) khi điều kiện thực hiện chuyển từ OFF ở chu kỳ quét trước sang ON ở chu kỳ quét lần này. Sau đó bit lại trở về trạng thái OFF.
- **DIFD(14)** : Lệnh này sẽ bật bit đi kèm lên 1 trong vòng một chu kỳ quét (scan/cycle) khi điều kiện thực hiện chuyển từ ON ở chu kỳ quét trước sang OFF ở chu kỳ quét lần này. Sau đó bit lại trở về trạng thái OFF.

Ký hiệu





Ví dụ: Khi bit 000.00 chuyển từ OFF ở chu kỳ quét trước lên ON ở chu kỳ quét hiện hành, bit 002.00 sẽ được bật lên ON trong vòng một chu kỳ. ở chu kỳ quét sau, bit 002.00 lại được quay trở về OFF.




```
00000   LD          000.00
00001   DIFU        002.00
00002   LD          002.00
00003   OR          100.00
00004   AND NOT     000.01
00005   OUT         100.00
00006   END
```

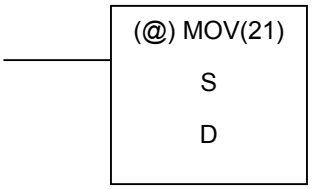
2.6.4) Lệnh copy dữ liệu MOVE - MOV(21)

Ký hiệu

S: Source word
IR, SR, AR, DM, HR, TC, LR, #

Các địa chỉ bộ nhớ truy cập ở dạng word hoặc hằng số (#)

D: Destination word
IR, SR, AR, DM, HR, LR



S = Là địa chỉ của word nguồn (Source word) hoặc một hằng số (# là ký hiệu của một hằng số, ví dụ #155,...được nhập vào ngay khi lập trình)

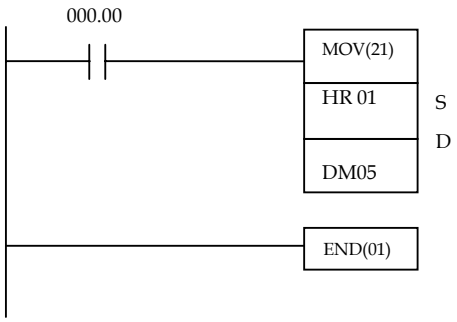
D = Là địa chỉ của word đích (Destination word)

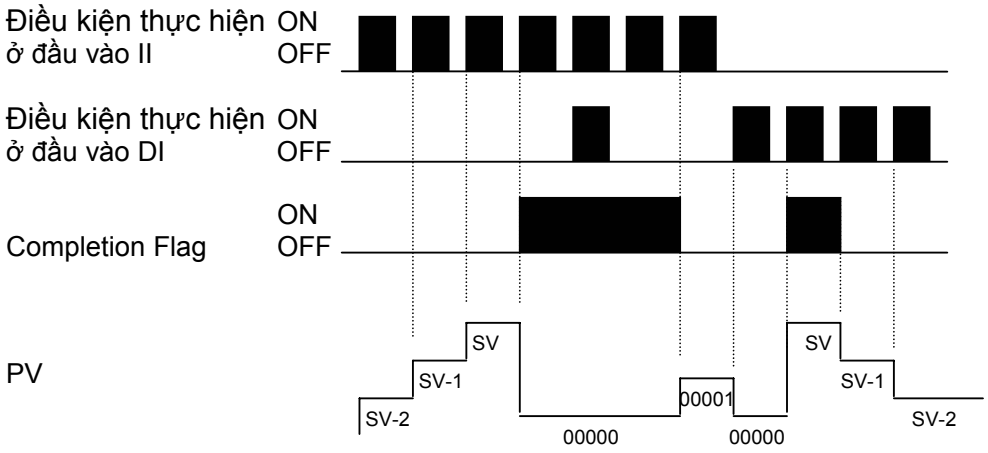
Khi lệnh MOV(21) có điều kiện thực hiện là ON, lệnh này sẽ copy hằng số hoặc nội dung của word có địa chỉ chỉ định bởi S sang word có địa chỉ chỉ định bởi D. Nội dung của word nguồn S không thay đổi khi thực hiện lệnh này. Ví dụ: Khi Source word là 000, còn Destination word là 100

Source word CH 000				**Destination word CH 100**		
00000		1	→	100.00		1
00001		1	→	100.01		1
00002		0	→	100.02		0
00003		1	→	100.03		1
00004		1		100.04		1
00005		0	.	100.05		0
00006		0	.	100.06		0
00007		1	.	100.07		1
00008		1		100.08		1
00009		1		100.09		1
00010		1		100.10		1
00011		0		100.11		0
00012		0		100.12		0
00013		0	→	100.13		0
00014		0	→	100.14		0
00015		1	→	100.15		1

Ví dụ:

Ở ví dụ dưới đây, địa chỉ word nguồn là S = HR01 (và nội dung của word này là giá trị 1500) còn địa chỉ của word đích là D = LR 05. Khi bit 000.00 lên ON, lệnh MOV(21) sẽ copy nội dung của HR01 (tức giá trị 1500) sang word LR05.





2.6.5) Bộ đếm lên xuống - Reversible Counter CNTR (FUN 12) (hay còn gọi là UP/DOWN Counter)

Chú ý : Mỗi bộ counter và timer có một số duy nhất từ 0 đến 127 và không được phép dùng trùng lặp trong lệnh đếm/timer khác của chương trình.

Số của bộ đếm và timer có 2 cách dùng như sau :

- Khi dùng như một bit, nó được dùng làm cờ báo đã đếm xong (completion flag).
- Khi dùng như một word, nó được dùng để truy cập giá trị đếm hiện tại (PV).

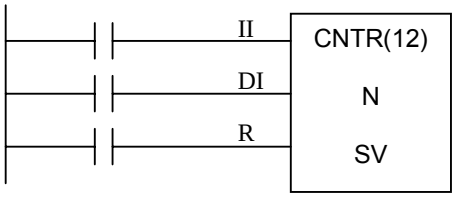
CNTR là một bộ đếm có thể đếm theo hai chiều tăng - giảm:

- Bộ đếm sẽ tăng giá trị của PV (Present Value) lên 1 mỗi khi đầu vào II (Increment Input) chuyển từ OFF lên ON.
- Bộ đếm sẽ giảm giá trị của PV (Present Value) đi 1 mỗi khi đầu vào DI (Decrement Input) chuyển từ OFF lên ON. Khi bộ đếm giảm đến 0, giá trị hiện tại của PV được gán cho SV và cờ báo hoàn thành (completion flag - chính là bit CNTR n với n = số của counter) sẽ lên ON cho đến khi bộ đếm lại giảm tiếp.
- Bộ đếm sẽ reset PV về 0 khi đầu vào Reset Input (R) chuyển từ OFF lên ON.

- Khi PV bằng với giá trị đặt SV (Set Value), PV được reset về 0 và cờ báo hoàn thành sẽ bật lên ON cho đến khi bộ đếm lại tiếp tục đếm tăng.
- Khi cả II và DI đều cùng chuyển từ OFF lên ON, bộ đếm vẫn giữ nguyên giá trị.

Ký hiệu

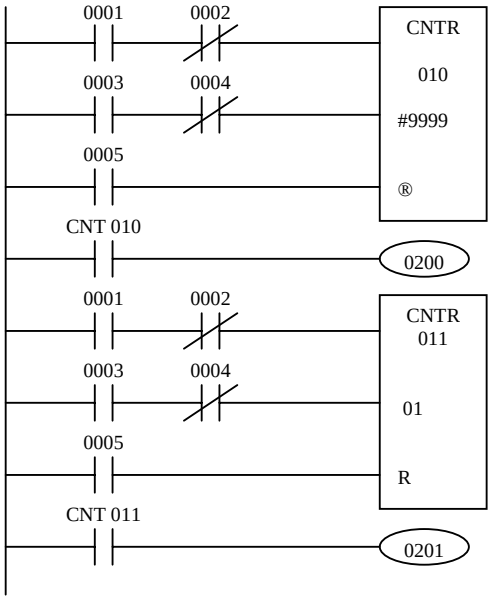
N : số của counter # (0-127)
SV: Set Value (word, BCD)
IR, SR, AR, DM, HR, LR, #



- II : Đầu váo đếm tăng
- DI : Đầu váo đếm giảm
- R : đầu vào reset giá trị PV
- SV : Giá trị đặt trước

Ví dụ minh hoạ Bộ đếm tăng giảm (UP/DOWN counter)

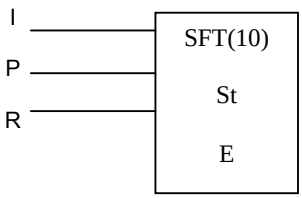
Ladder Diagram



Mnemonic Code

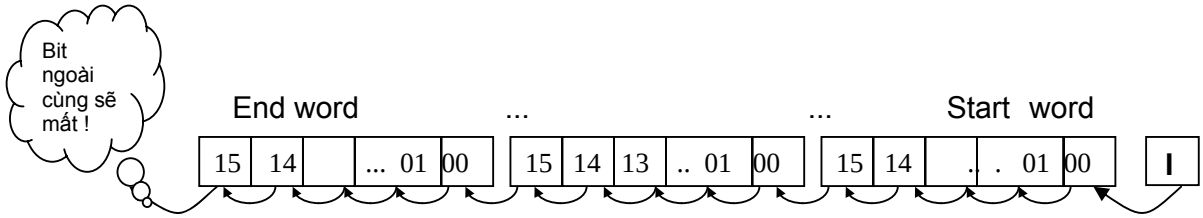
Đ. chỉ	Lệnh	Th. số	
0200	LD		0001
0201	AND-NOT		0002
0202	LD		0003
0203	AND-NOT		0004
0204	LD		0005
0205	CNTR(12)		010
		#	9999
0206	LD	CNT	010
0207	OUT		0200
0208	LD		0001
0209	AND-NOT		0002
0210	LD		0003
0211	AND-NOT		0004
0212	LD		0005
0213	CNTR(12)		011
			01
0214	LD	CNT	011
0215	OUT		0201

2.6.6) Thanh ghi dịch - SHIFT REGISTER - SFT(10)



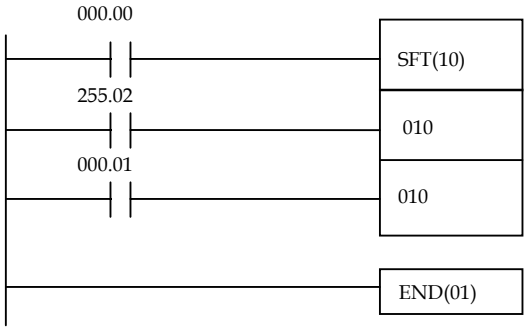
St = Word đầu tiên của thanh ghi dịch
E = Word cuối của thanh ghi dịch
I = Bit dịch (Input bit)
P = Bit xung nhịp ((Shifting) Pulse Input)
R = Đầu vào xoá (Reset Input)

Thanh ghi dịch được định nghĩa là các word bắt đầu từ Word đầu tiên St cho đến Word cuối E (địa chỉ Word cuối phải > Word đầu).



Ví dụ minh hoạ:

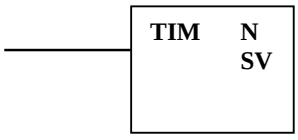
Ở ví dụ dưới đây, ta có 1 thanh ghi dịch dài 1 word (St= 010, E =010) tại địa chỉ 010. Lệnh SFT(10) sẽ dịch các bit của của thanh ghi dịch sang bên trái một vị trí bit và bit 000.00 được dịch vào bit ngoài cùng bên phải (tức bit 010.00) của thanh ghi này mỗi khi bit 255.02 chuyển từ OFF lên ON. Bit 255.02 này là một bit xung nhịp 1 giây do đó thanh ghi dịch sẽ được dịch sang trái, bit ngoài cùng bên trái (tức bit 010.15) sẽ mất mỗi giây một lần. Khi bit 000.01 (đầu vào Reset) lên ON, nội dung của thanh ghi dịch sẽ được reset về 0 (các bit đều bị reset về 0).



Đ. chỉ	Lệnh	Th. số
00000	LD	00000
00001	LD	25502
00002	LD	00001
00003	SFT(10)	
		010
		010
00004	END(01)	

2.6.7) Rơle thời gian (TIMER) - TIM

Ký hiệu



N: Số của Timer
Hằng số (#)

SV: Set Value (Word, BCD)
IR, SR, AR, DM, HR, LR, #

Các địa chỉ bộ nhớ truy cập ở dạng word hoặc hằng số (#)

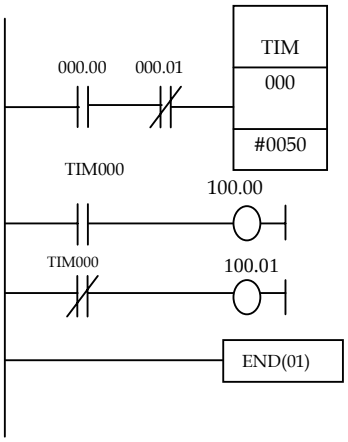
N = Số của timer hiện dùng (Timer Number) (số hợp lệ là từ 000 - 127)
SV = Giá trị đặt trước Set Value tính theo đơn vị là 0,1s (SV phải ở dạng số BCD hoặc chỉ đến một Word có chứa giá trị BCD). Giá trị của SV phải nằm trong khoảng từ 0000 - 9999 (0 - 999,9 Giây.)

Khi đầu vào điều kiện thực thi của hàm TIM là ON, hàm TIM sẽ đếm giảm thời gian từ giá trị thời gian đặt trước SV đến khi bằng 0 thì completion flag (TIM

n) lên ON. Completion flag sẽ vẫn ở ON cho đến khi bị reset bởi đầu vào điều kiện thực hiện về OFF.

Ví dụ: Timer số 000 (TIM000) có đầu vào điều kiện thực hiện do hai bit 000.00 và 000.01 quyết định. Khi bit 000.00 là ON và bit 000.01 là OFF, timer bắt đầu đếm giảm thời gian PV theo từng đơn vị là 0,1 giây từ giá trị đặt trước SV là 5,0 giây. Khi giá trị thời gian hiện tại PV về đến 0, cờ completion flag TIM000 sẽ lên ON và bật bit 010.00 lên ON còn bit 010.01 về OFF.

Ladder Diagram

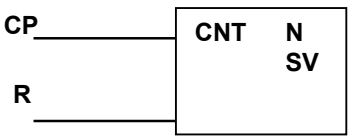


Mnemonic Code

```
00000   LD          000. 00
00001   AND NOT     000. 01
00002   TIM         000     #0050
00003   LD          TIM000
00004   OUT         100.00
00005   LD NOT      TIM000
00006   OUT         100.00
00007   END
```

2.6.8) Bộ đếm giảm (COUNTER) - CNT

Lúc khởi đầu giá trị PV được đặt bằng SV (Set Value). Mỗi khi đầu vào xung đếm CP chuyển từ OFF lên ON, giá trị đếm hiện tại PV (Present Value) sẽ giảm một đơn vị. Khi PV giảm đến 0, cờ báo kết thúc sẽ lên ON và sẽ ở ON cho đến khi counter được reset bởi đầu vào R (Reset).



N: Số của bộ đếm
Hằng số (#)

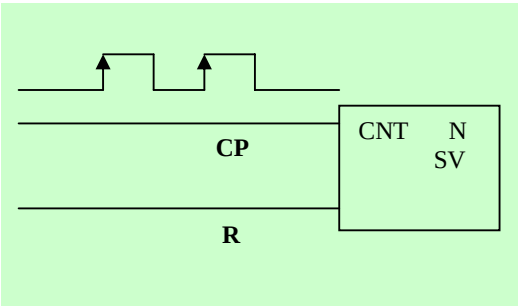
SV: Set Value (Word, BCD)
IR, SR, AR, DM, HR, LR, #

Các địa chỉ bộ nhớ truy cập ở dạng word hoặc hằng số (#)

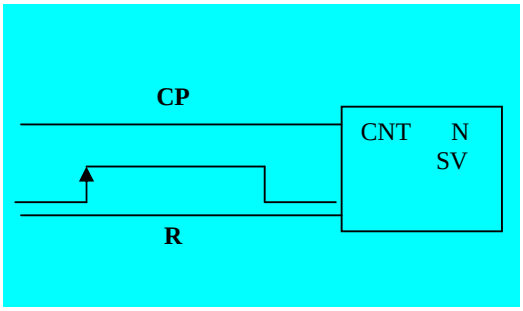
- N : Số của bộ đếm (từ 000 đến 127)
- SV : Giá trị đặt (từ 0 đến 9999) và phải ở dạng BCD
- CP : Đầu vào xung đếm
- R : Đầu vào reset

Ví dụ:

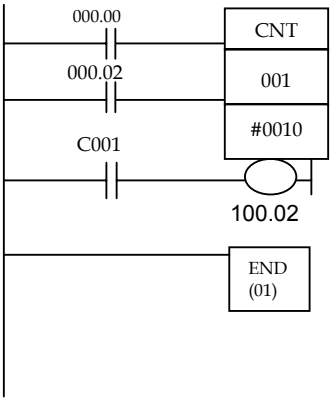
Bộ đếm đang đếm



Bộ đếm bị reset



Ví dụ: Giá trị hiện hành (PV) của bộ đếm CNT001 sẽ giảm từ giá trị SV khi đầu vào Input 00000 chuyển từ OFF lên ON. Khi số lần chuyển từ OFF lên ON của input 00000 là 10 lần (bằng với SV=10), cờ CNT001 sẽ lên ON và do đó bật đầu ra 100.02 lên ON. Cờ CNT001 và PV của bộ đếm sẽ bị reset khi đầu vào input 00002 lên ON.




```
00000   LD    000.00
00001   LD    000.02
00002   CNT   001
              #0010
00003   LD    CNT001
00004   OUT   100.02
00005   END
```

2.6.9) Ví dụ về ứng dụng COUNTER và TIMER

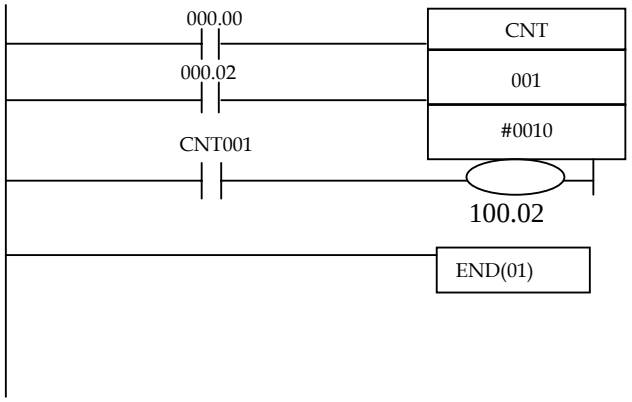
Ví dụ 1 Mở rộng khả năng đếm của counter

Một Photo Switch được dùng để phát hiện sản phẩm và đưa vào đầu vào của counter. Yêu cầu cần phải đếm được 20.000 sản phẩm thì cho ra đèn OUTPUT LAMP (tuy nhiên bộ đếm CNT chuẩn chỉ cho phép đếm tới 9.999).

I/O	
PHOTO SWITCH	000.00
OUTPUT LAMP	010.00
PB RESET	000.01

Ladder Mnemonic Code

Đ. chỉ	Lệnh	Th.số
00000	LD	00000
00001	LD	CNT 001
00002	CNT	001
		#0100
00003	LD	CNT 001
00004	LD	00001
00005	CNT	002
		#0200
00006	LD	CNT 002
00007	OUT	100.02
00010	END(01)	




```
00000   LD      000.00
00001   LD      000.02
00002   CNT     001
                #0010
00003   LD      CNT001
00004   OUT     100.02
00005   END
```

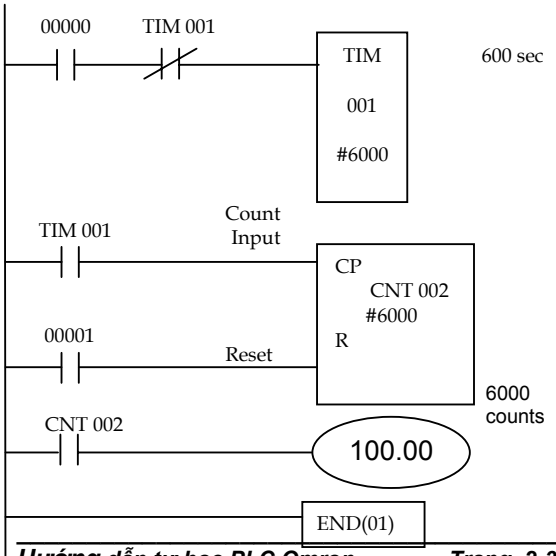
Ví dụ 2 Kéo dài thời gian trễ của timer lên 1.000 giờ

TIM chuẩn chỉ cho phép đặt thời gian tới 999,9 giây. Chương trình sau đây cho phép kéo dài khả năng của TIM lên 1.000 giờ.

I/O

PB START	000.00
PB RESET	000.01
VALUE LUBRICATE	100.00

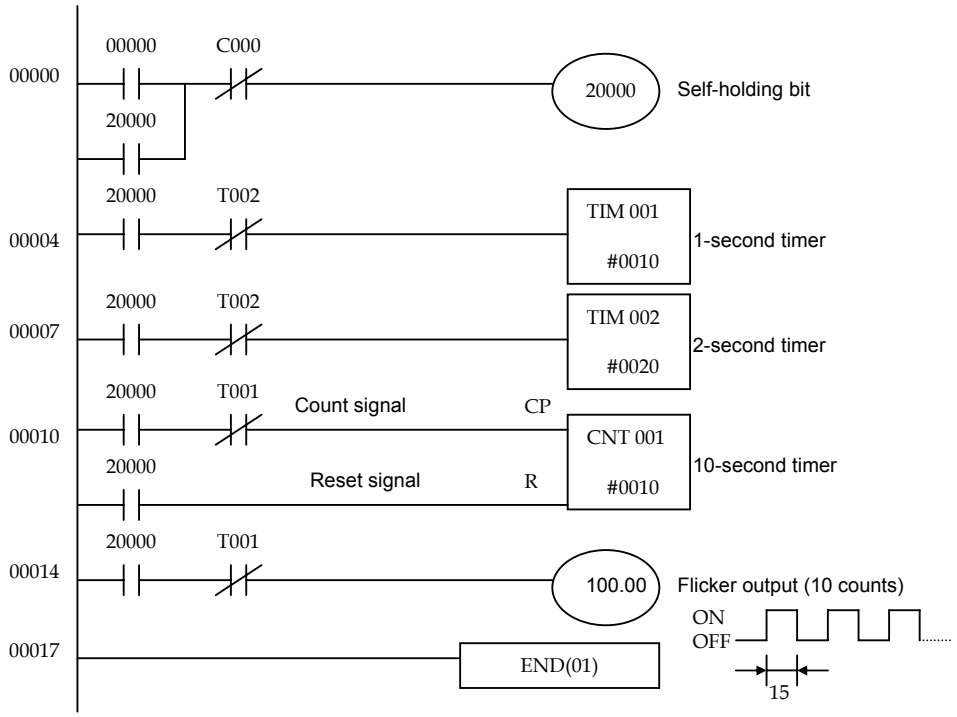
Ladder



Mnemonic Code

Đ. chỉ	Lệnh	Th.số
00000	LD	00000
00001	AND-NOT	TIM 001
00002	TIM	001
		# 6000
00003	LD	TIM 001
00004	LD	00001
00005	CNT	002
		# 6000
00006	LD	CNT 002
00007	OUT	100.00
00008	END (01)	

Ví *dụ 3* Chương trình này sẽ làm nhấp nháy (flicker) đầu ra 100.00 (bật 1 giây, tắt 1 giây) ON/OFF 10 lần sau khi bit 000.00 lên ON.

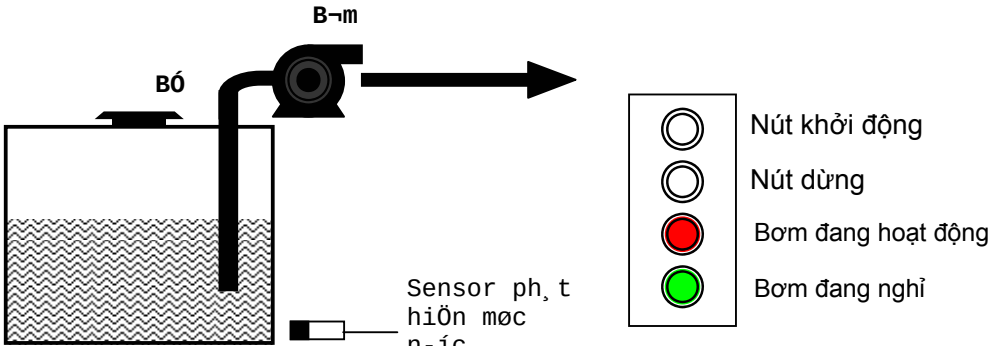


Đ. chỉ	Lệnh	Th. số		Chú thích
00000	LD		00000	(1) Self-holding bit
00001	OR		20000	
00002	AND NOT	C*	000	
00003	OUT		20000	
00004	LD		20000	(2) 1-Second timer
00005	AND NOT	T**	002	
00006	TIM		001	
		#	0010	
00007	LD		20000	(3) 2-Second timer
00008	AND NOT	T	002	
000009	TIM		002	
		#	0020	
00010	LD		20000	(4) 10-count counter
00011	AND	T	001	
00012	LD NOT		20000	
00013	CNT		000	
		#	0010	
00014	LD		20000	(5) Flicker output (10 counts)
00015	AND NOT	T	001	
00016	OUT		100.00	
00017	END(01)		- - -	(6) END(01) Lệnh

* : C= Counter
** : T = Timer

Ví dụ 4: Một hệ thống điều khiển máy bơm đơn giản

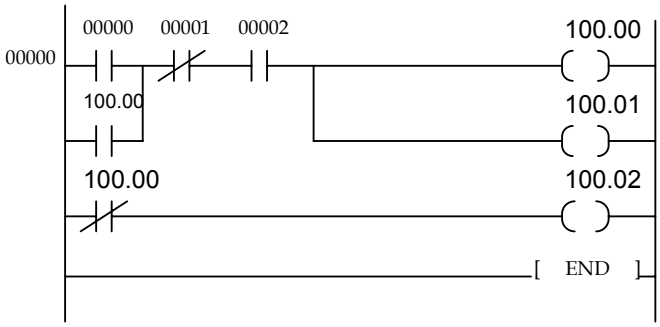
Khi nút Khởi động START được bấm, bơm sẽ kiểm tra mức nước xem có thể bơm được không qua tín hiệu từ sensor đo mức nước, nếu mức nước đạt thì bơm sẽ bơm liên tục cả khi nút Khởi động đã nhả. Bơm sẽ dừng khi nút dừng STOP được bấm hoặc khi mức nước xuống thấp quá. Kèm theo là các đèn chỉ thị tình trạng bơm.



Các đầu vào ra (I/O)

I/O	Địa chỉ trên PLC	Chức năng
INPUT	00000	Nút khởi động
	00001	Nút dừng
	00002	Sensor phát hiện mức nước
OUTPUT	100.00	Đầu ra điều khiển bơm
	100.01	Đèn báo bơm đang chạy
	100.02	Đèn báo bơm đang nghỉ

Chương trình dạng sơ đồ bậc thang



Chương trình dạng Mnemonic code:

Đ. chỉ	Lệnh	Th. số
00000	LD	00000
00001	OR	01000
00002	AND NOT	00001
00003	AND	00002
00004	OUT	100.00
00005	OUT	100.01
00006	LD NOT	100.00
00007	OUT	100.02
00008	END (01)	

CHƯƠNG 3

Phần mềm **CX-PROGRAMMER**

Vài nét về bộ phần mềm CX-ONE

CX-ONE là 1 bộ phần mềm được tích hợp chặt chẽ nhằm đáp ứng những yêu cầu ngày càng cao trong tự động hóa công nghiệp và hỗ trợ các thiết bị rất đa dạng của OMRON. Với các phần mềm này, người sử dụng có trong tay những công cụ mạnh, sử dụng dễ dàng và liên tục được cập nhật, cải tiến.

	**CX-Programmer**	CX-Programmer cung cấp 1 nền tảng chung cho phát triển chương trình cho tất cả các loại PLC Omron từ các loại micro PLC đến những loại PLC Duplex cao cấp
	CX-**Compolet**	Sysmac Compolet cung cấp cho các nhà phát triển phần mềm các thành phần để trợ giúp việc phát triển các phần mềm kết nối với các bộ điều khiển của OMRON dùng các công cụ như Microsoft Visual Studio.Net.
	**CX-Reporter**	CX-Reporter cho phép người sử dụng đọc và ghi dữ liệu từ PLC bằng Microsoft Excel mà không cần phải lập trình
	CX-Integrator	CX-Integrator giúp cấu hình các mạng công nghiệp kết nối dùng PLC như Controller Link, DeviceNet, CompoNet, CompoWay, Ethernet, bao gồm cả các chức năng Routing Table Component và Data Link Component
	**CX-Process Tool**	CX-Process Tools là công cụ đi kèm với khối module PLC Loop Control Board/Unit của OMRON, cho phép tạo và thử các quy trình điều khiển tuần tự & vòng cũng như các khối chức năng cho khối này
	**CX-Motion**	Cx-Motion giúp việc đặt thông số, theo dõi và lập trình với ngôn ngữ G-Code cho các bộ điều khiển chuyển động loại CS1-MC series của OMRON trở nên dễ dàng và rất trực quan.
	**CX-Position**	Cx-Position trợ giúp đặt thông số, theo dõi và lập trình bằng ngôn ngữ G-Code cho các bộ điều khiển chuyển động loại CJ1/CS1-NC series của OMRON.

	**CX-Simulator**	Cx-Simulatior là phần mềm mô phỏng các loại PLC CS1/CJ1 Series của OMRON. Nó cho phép mô phỏng hoạt động của PLC ngay trên máy tính mà không cần phải tải phần mềm vào phần cứng PLC, vì vậy rất thích hợp cho việc kiểm tra & sửa lỗi.
	**CX-Protocol**	Cx-Protocol giúp xây dựng các chương trình kết nối với các thiết bị của hãng thứ ba qua giao tiếp nối tiếp bằng các card truyền thông của họ PLC CS1/CJ1 & các họ PLC khác. Sau đó việc thực hiện truyền thông sẽ thực hiện bằng lệnh PMCR trong ngôn ngữ bậc thang.
	**CX-Profibus**	CX-Profibus trợ giúp việc đặt cấu hình, chỉnh sửa thông số, chẩn đoán & bảo trì mạng Profibus
	CX-Thermo	Phần mềm dùng để đặt thông số cho các thiết bị công nghiệp của Omron như bộ điều khiển nhiệt độ
	CX-Designer	Phần mềm thiết kế các trang màn hình giao diện cho màn hình loại series NS

CX-Programmer là phần mềm trung tâm của gói phần mềm trên. Không chỉ dùng để lập trình cho PLC, CX-Programmer còn là công cụ để các kỹ sư quản lý 1 dự án tự động hóa với PLC làm bộ não hệ thống.
Các chức năng chính của CX-Programmer bao gồm:

- Tạo và quản lý các dự án (project) tự động hóa (tức các chương trình)
- Kết nối với PLC qua nhiều đường giao tiếp
- Cho phép thực hiện các thao tác chỉnh sửa & theo dõi khi đang online (như force set/reset, online edit, monitoring,..)
- Đặt thông số hoạt động cho PLC
- Cấu hình đường truyền mạng
- Hỗ trợ nhiều chương trình, nhiều PLC trong 1 cùng project & nhiều section trong 1 chương trình

CX-Programmer hiện có 3 phiên bản chính:
- Bản Junior 2.1: Bản này chỉ hỗ trợ các loại PLC micro của OMRON như CPMx, SRM1. Hiện tại phiên bản này được cung cấp miễn phí cho các khách hàng mua PLC OMRON tại Việt nam.
- Bản Junior: Bản này chỉ hỗ trợ các loại PLC micro của OMRON như CP1L/ CP1H, CPMx, SRM1.
- Bản đầy đủ: Bản này hỗ trợ tất cả các loại PLC của OMRON, ngoài loại CPMx, SRM1 còn có các loại thông dụng khác như CQM1x, C200x, CS1, CJ1x.

CP1L/1H có thể được lập trình từ máy tính (PC) có chạy phần mềm CX-Programmer version 7.xx trở lên.

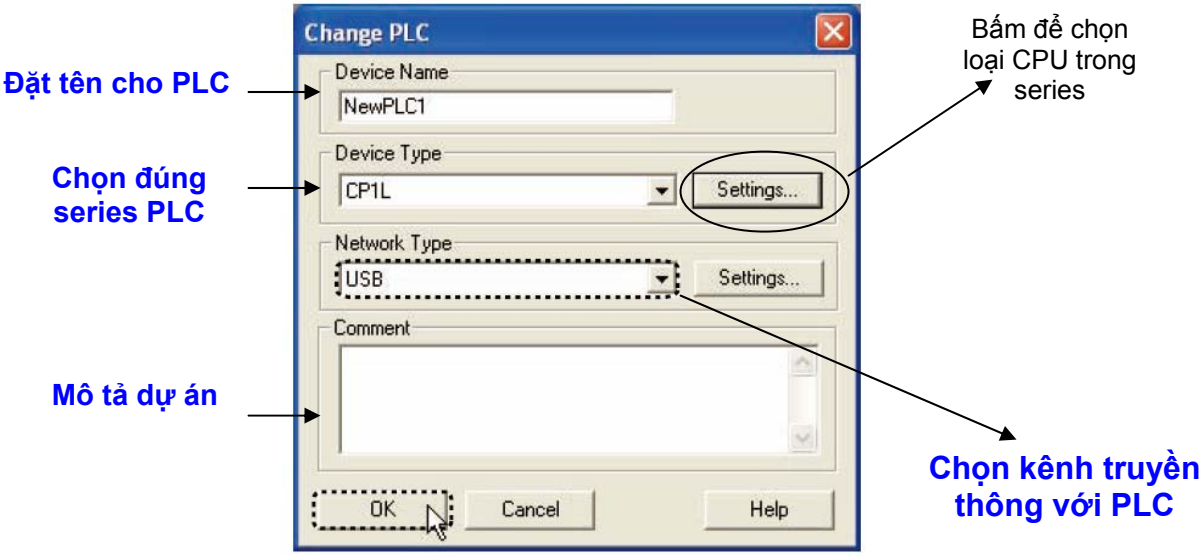
Phần tiếp theo xin giới thiệu từng bước về 1 số các thao tác cơ bản với CX-Programmer.
Các ký hiệu quy ước dùng trong tài liệu:

Chỉ thao tác bấm nút trái chuột

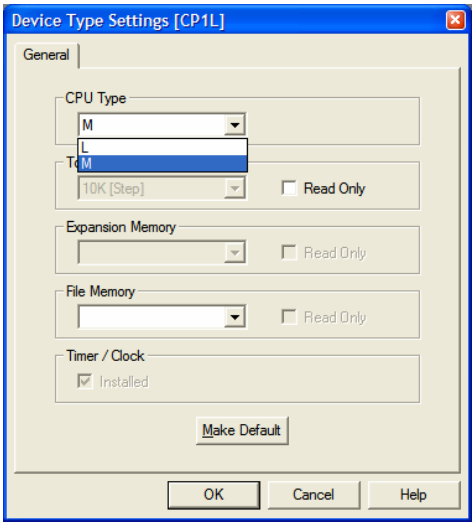
Chỉ thao tác bấm đúp nút trái chuột

Chỉ thao tác bấm nút phải chuột

Tao 1 project mới

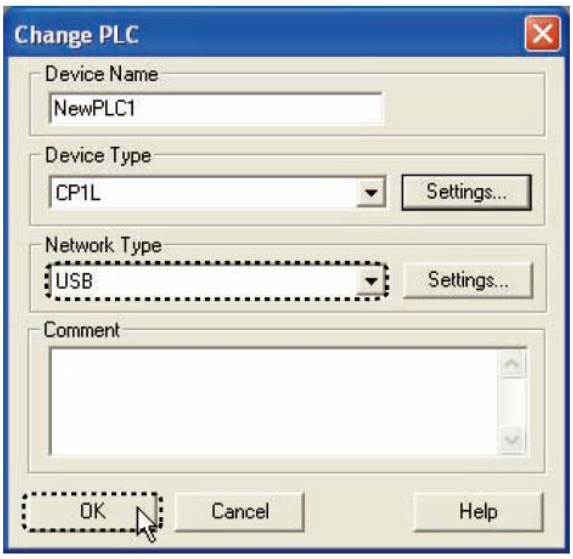


Chọn loại CPU



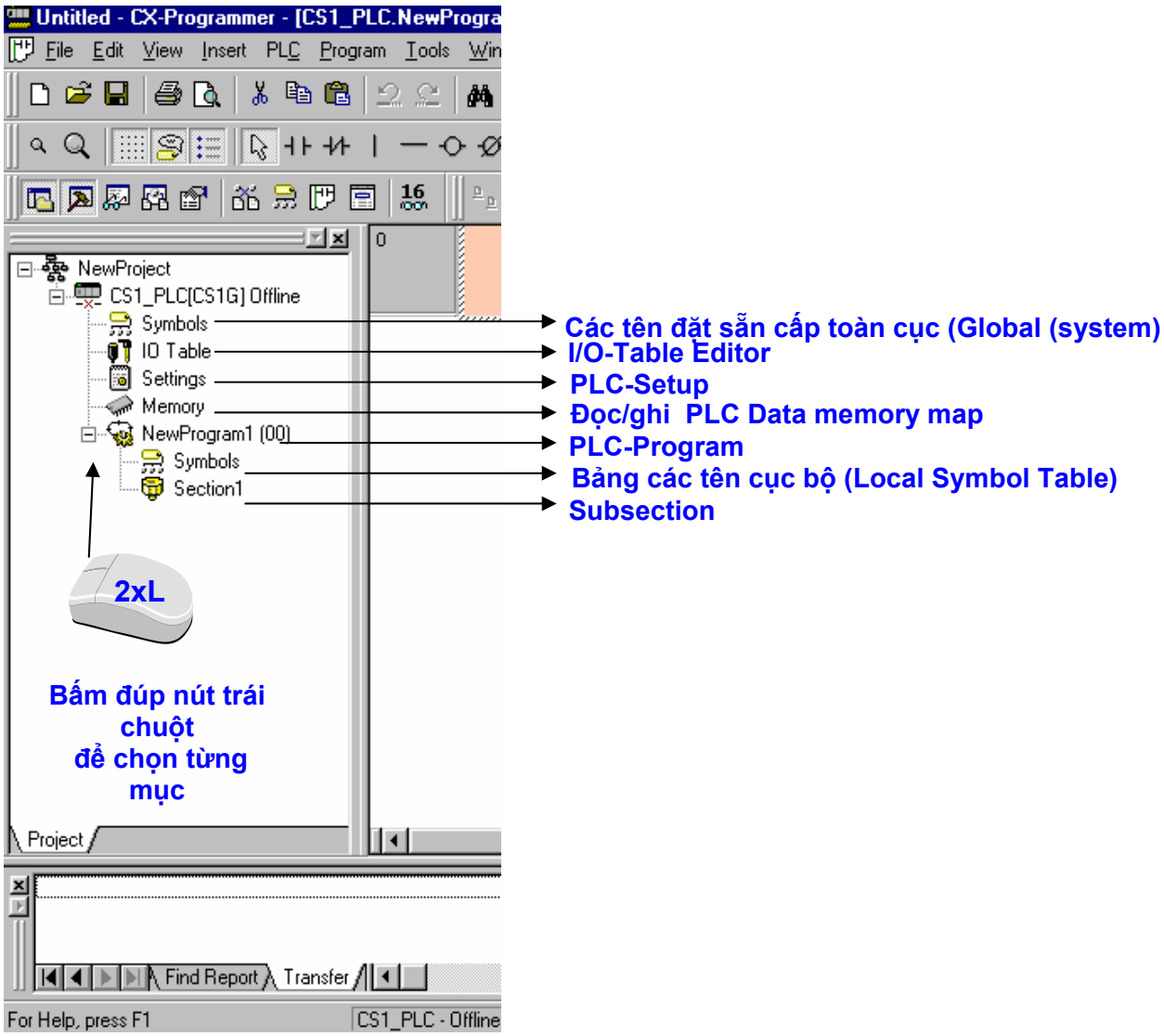
Với series CP1L, lựa chọn loại L hay M tùy theo model đang dùng. Các lựa chọn khác không cần thay đổi (để nguyên như mặc định)

Chọn kênh truyền thông



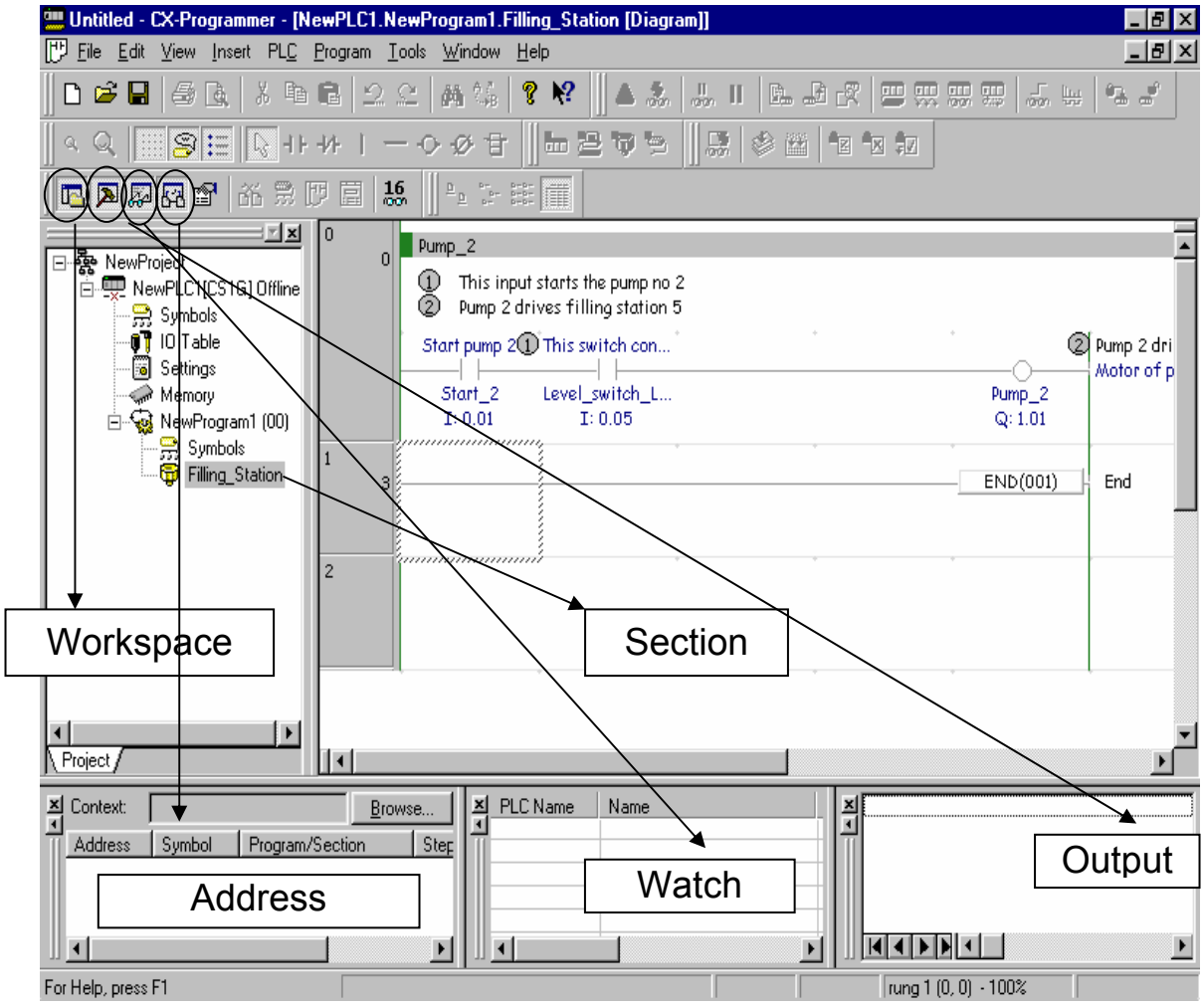
Các thông số này thường là không cần thay đổi vì các thông số mặc định đã được đặt sẵn phù hợp với loại PLC đang dùng. **Network Type** cần chọn là USB như hình trên đối với loại CP1L/CP1H khi dùng cáp USB để kết nối với PLC.
Trường hợp CX-Programmer không thể kết nối với PLC, hãy kiểm tra lại thông số này.

❖ **Các thành phần trên cửa sổ project:**



Các cửa sổ phụ trên màn hình giao diện của CX-Programmer

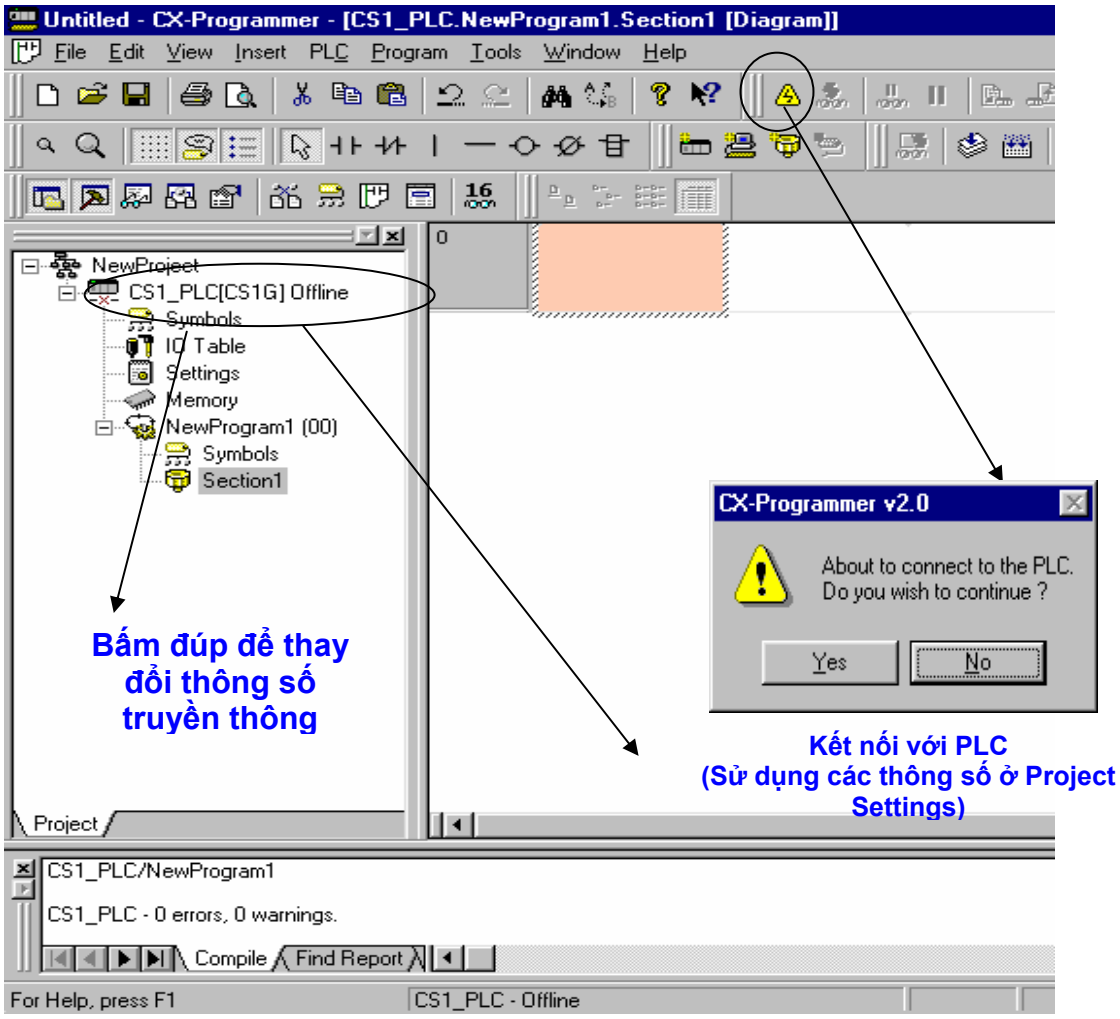
Trong quá trình làm việc với **CX-Programmer**, người sử dụng có thể bật hoặc tắt các cửa sổ phụ. Các cửa sổ này hiển thị các thông tin có liên quan đến các đối tượng & công việc đang được thực thi.



- Cửa sổ Workspace: là cửa sổ thường nằm bên trái màn hình & liệt kê các thông tin chính trong 1 chương trình như Symbol, Section, Settings, Memory...
- Cửa sổ Address Reference: cho phép quan sát việc sử dụng 1 địa chỉ bộ nhớ bất kỳ trong chương trình
- Cửa sổ Watch: Với cửa sổ này, người sử dụng có thể quan sát giá trị của 1 địa chỉ trong bộ nhớ cũng như thực hiện các thao tác thay đổi giá trị của chúng ngay từ CX-Programmer
- Cửa sổ Output: Các kết quả kiểm tra & biên dịch chương trình cùng các thông tin khác sẽ được hiển thị trên cửa sổ này.

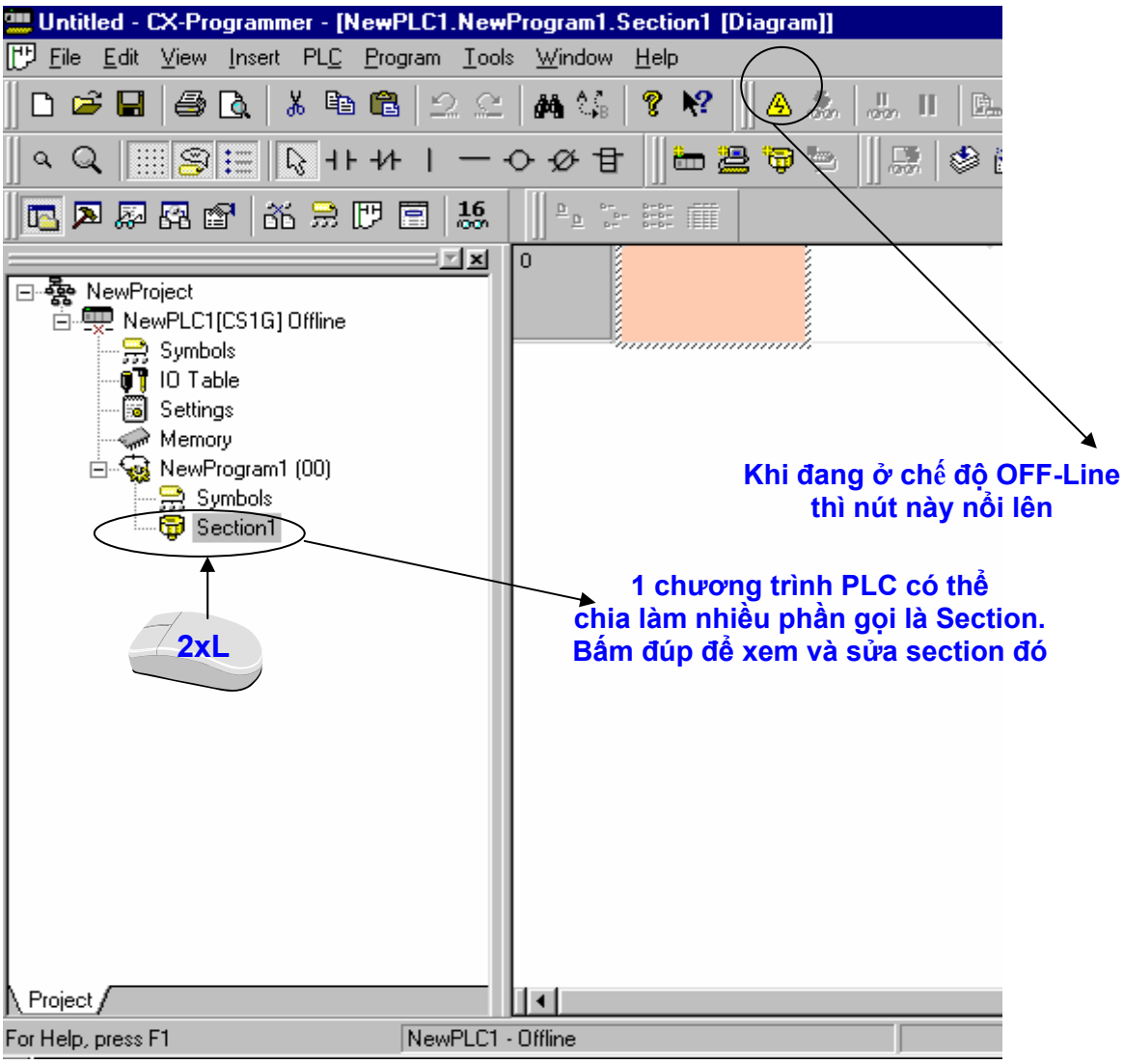
Kiểm tra kết nối (Communication) với PLC

Bấm vào nút Work Online để kết nối với PLC sau khi đã nối cáp giữa máy tính với PLC. Sau khi kết nối được thiết lập, CX-Programmer sẽ ở chế độ làm việc Online.

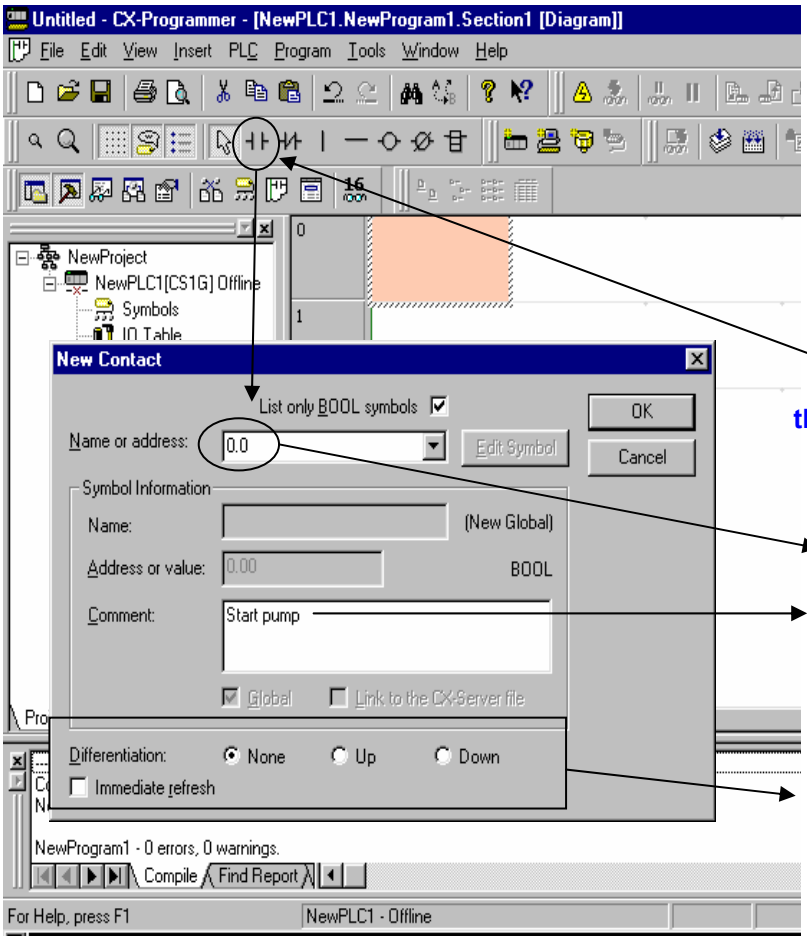


Bấm lại vào nút Work Online sẽ chuyển sang chế độ Offline để có thể sửa chương trình

Bấm đúp vào Section1 để hiển thị cửa sổ sửa chương trình bên phải



Thêm tiếp điểm



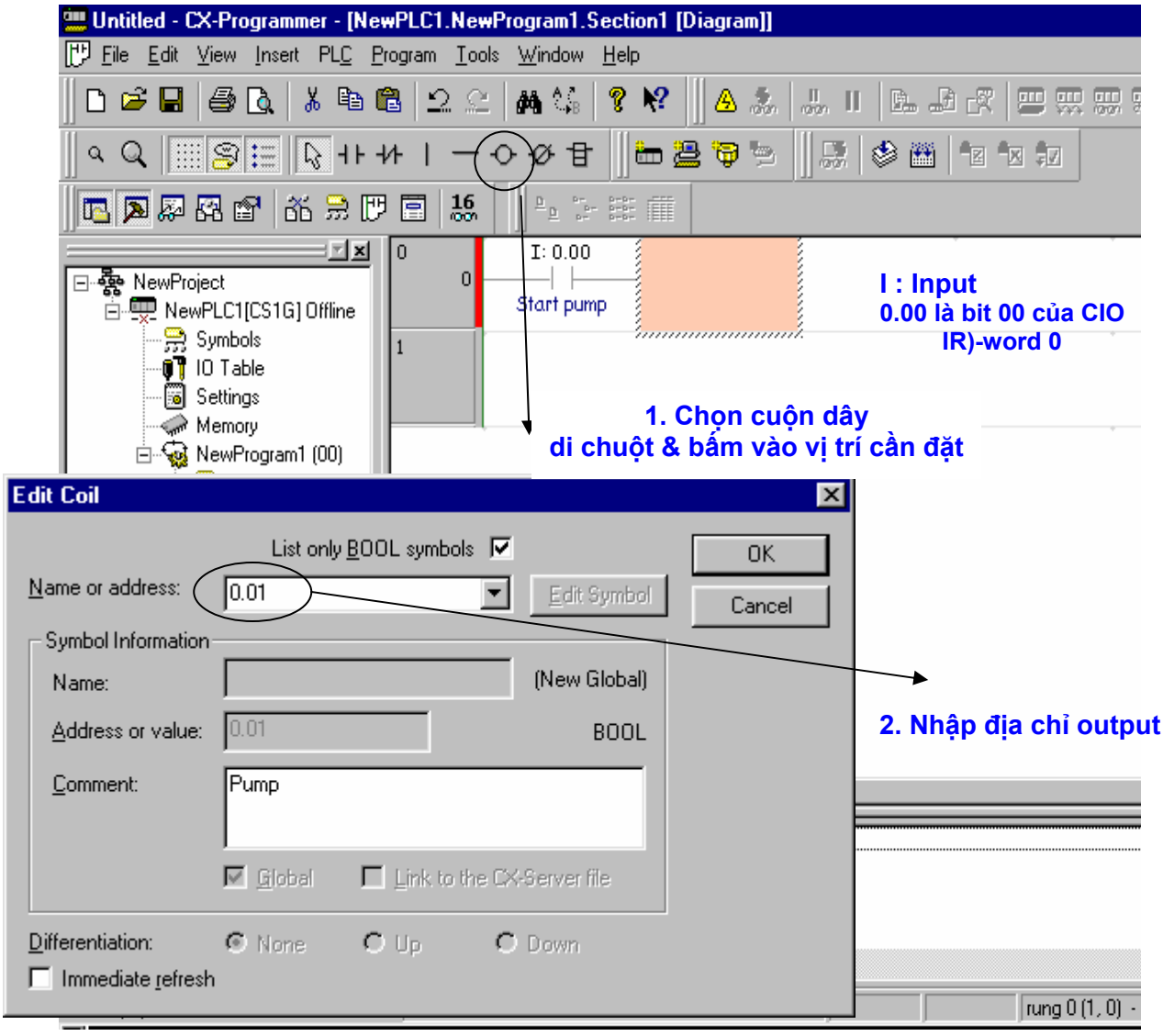
1. Chọn tiếp điểm thường mở, di chuột & bấm vào vị trí cần đặt trên section 1

2. Đánh địa chỉ của tiếp điểm

3. Thêm comment (tùy chọn)

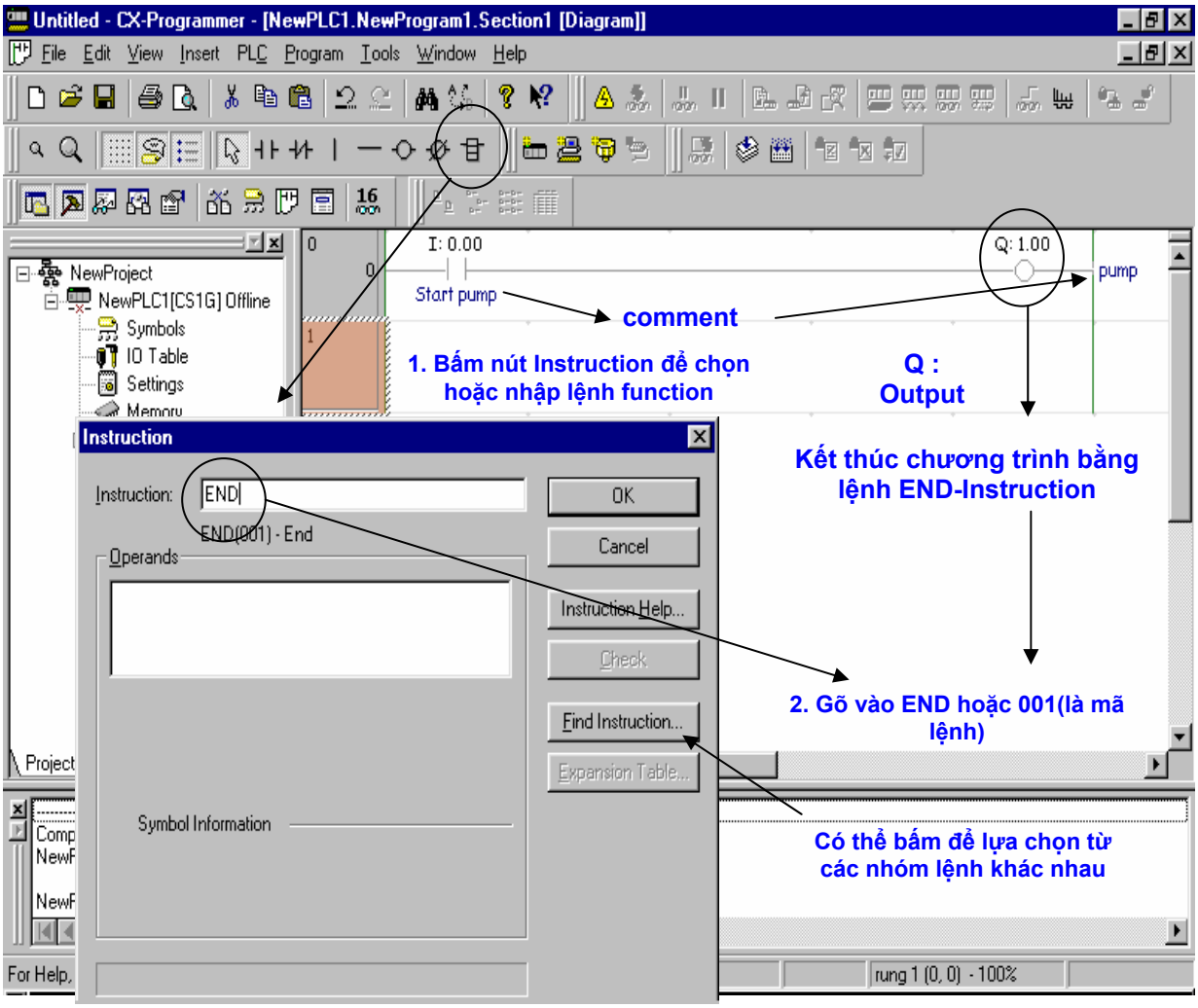
4. Hướng sườn tác động (tuỳ chọn)

Thêm cuộn dây



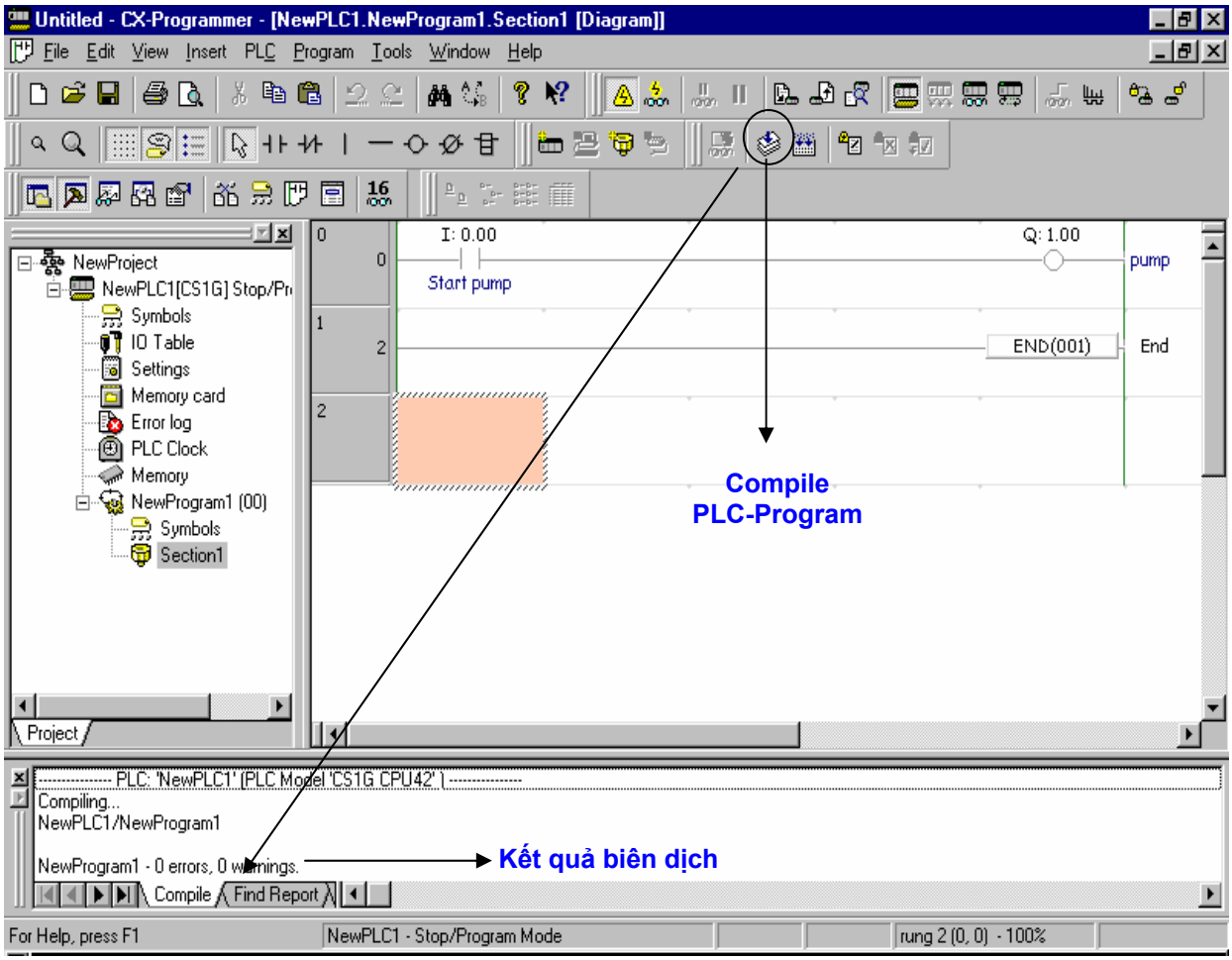
Thêm function

Mọi chương trình đều cần có ít nhất 1 lệnh End để đánh dấu điểm kết thúc của chương trình. Lệnh End và nhiều khối chức năng khác (function) có thể nhập vào dùng công cụ Instruction.



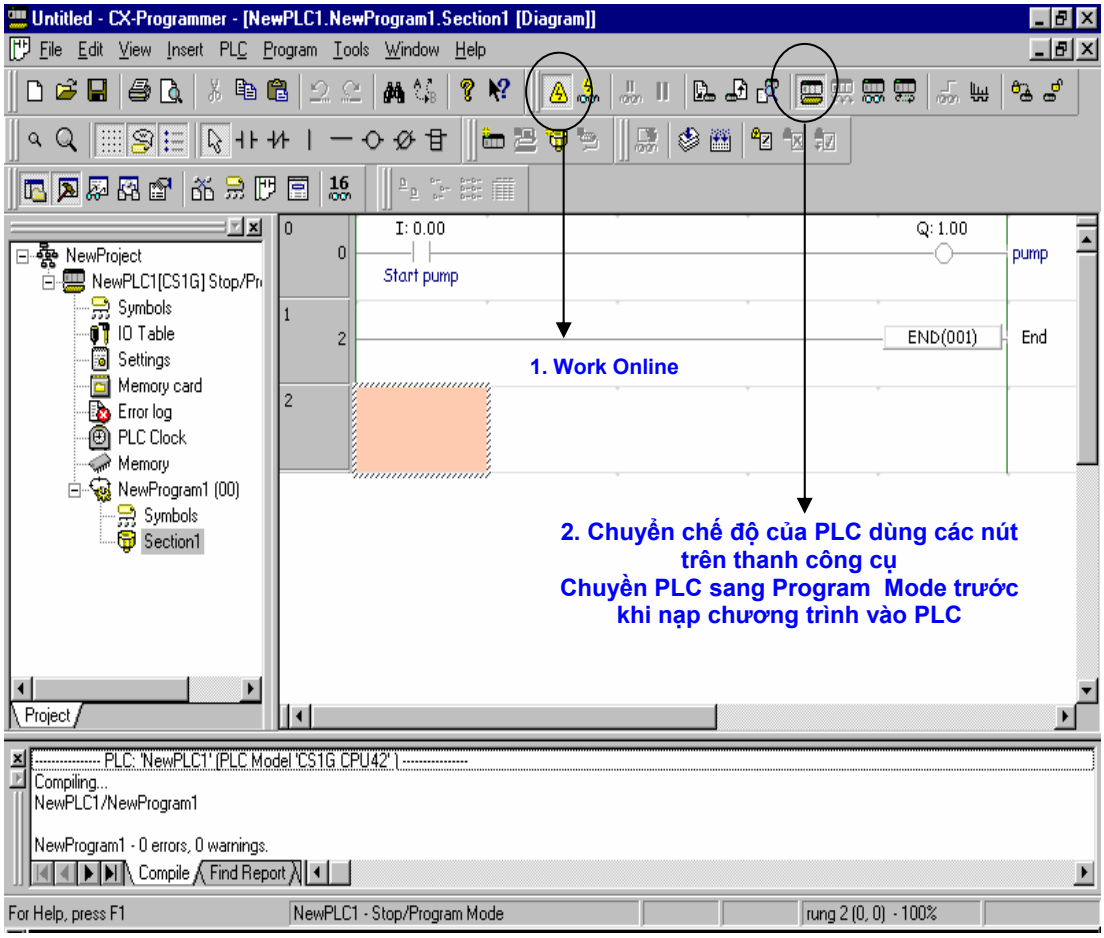
Kiểm tra & biên dịch chương trình

Việc biên dịch chương trình để nhằm phát hiện các lỗi do sai cú pháp, thiếu/thừa các phần tử,.. trong chương trình. Kết quả biên dịch được hiển thị trong tab compile của cửa sổ Ouput.



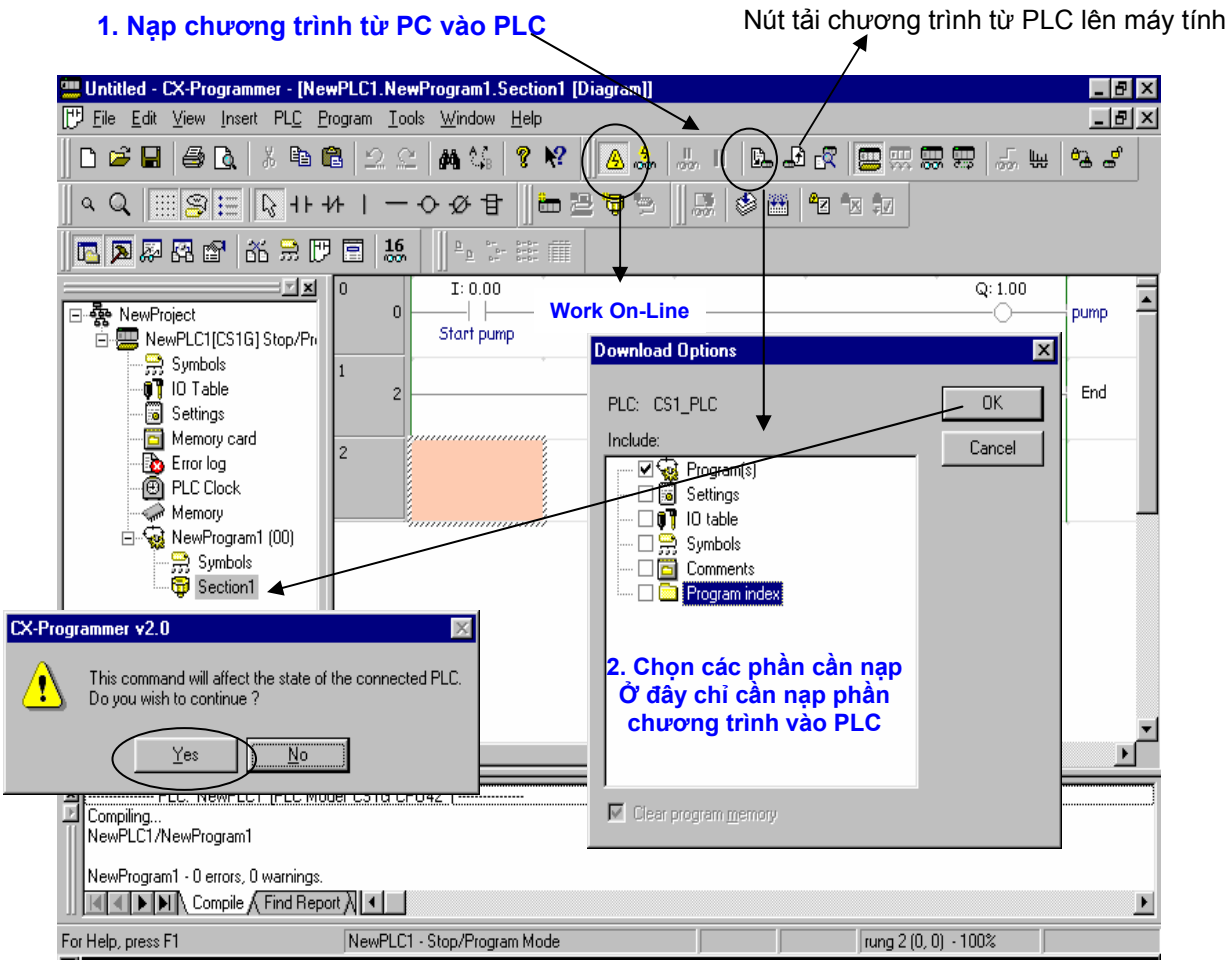
Bước tiếp theo chúng ta sẽ nạp chương trình đã viết vừa qua vào PLC. Về nguyên tắc, PLC cần chuyển sang Program Mode trước khi cho phép thay đổi nội dung chương trình PLC. Tuy vậy, ta có thể nạp chương trình vào PLC kể cả khi đang ở bất kỳ chế độ nào nhờ có các tính năng của CX-Programmer trợ giúp.

Bấm nút **Work Online** để kết nối với PLC, sau đó sử dụng các nút trên thanh công cụ để thay đổi chế độ chạy của PLC.

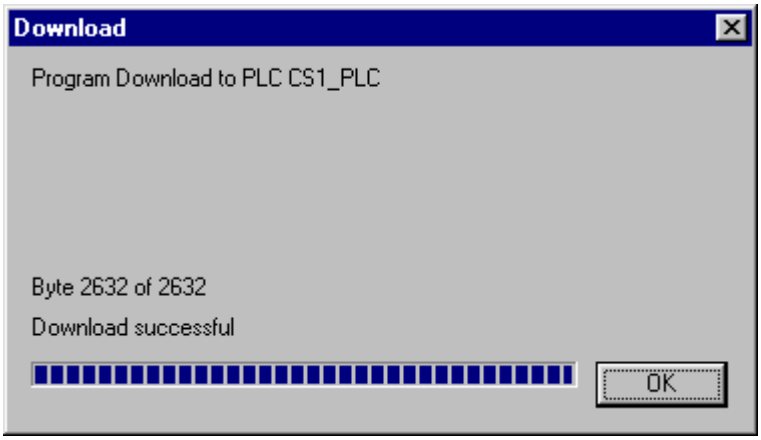


Khi đang online với PLC, các nút này cũng trực tiếp phản ánh chế độ làm việc hiện hành của PLC.

Nap (Download) chương trình vào PLC

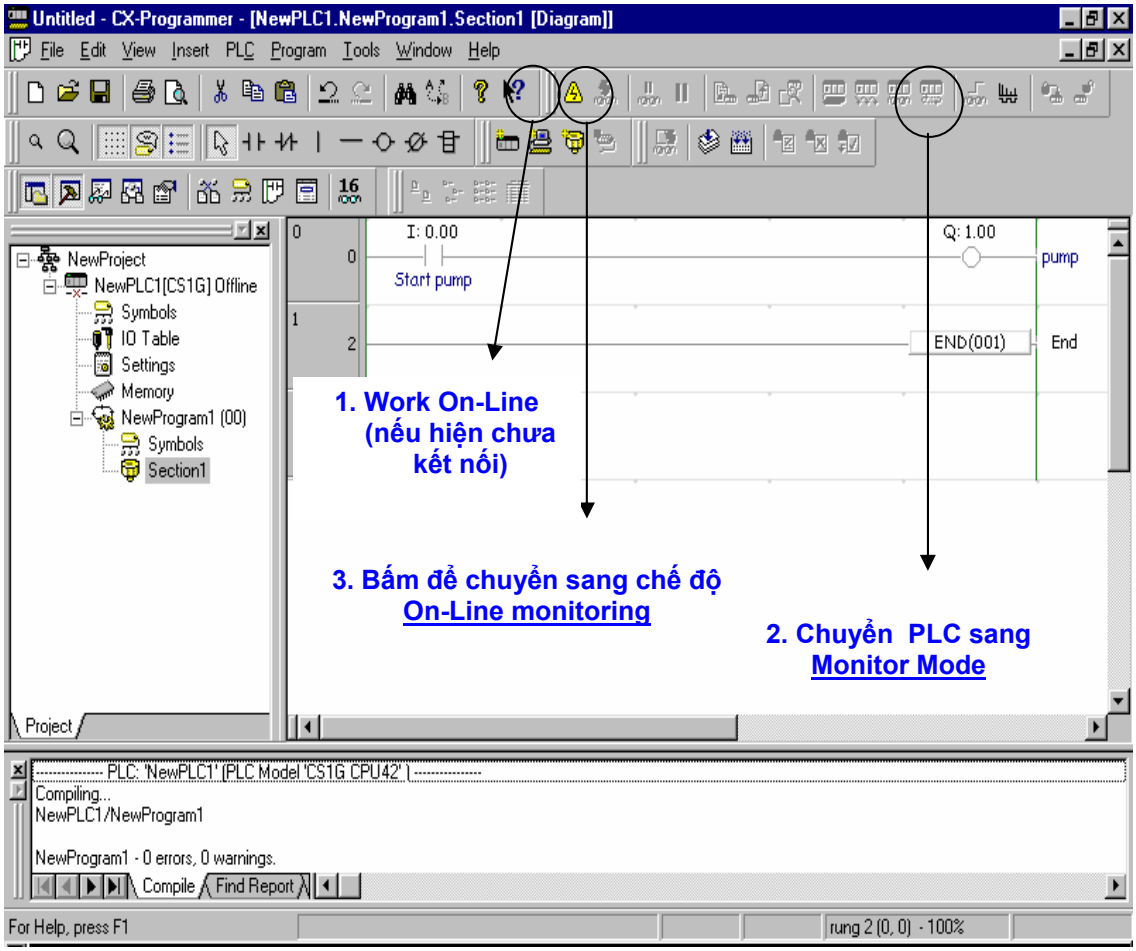


Việc nạp chương tình vào PLC cũng sẽ xóa nội dung hiện đang có trong PLC. Vì thế cần thận trọng xác nhận việc này trước khi tiếp tục.



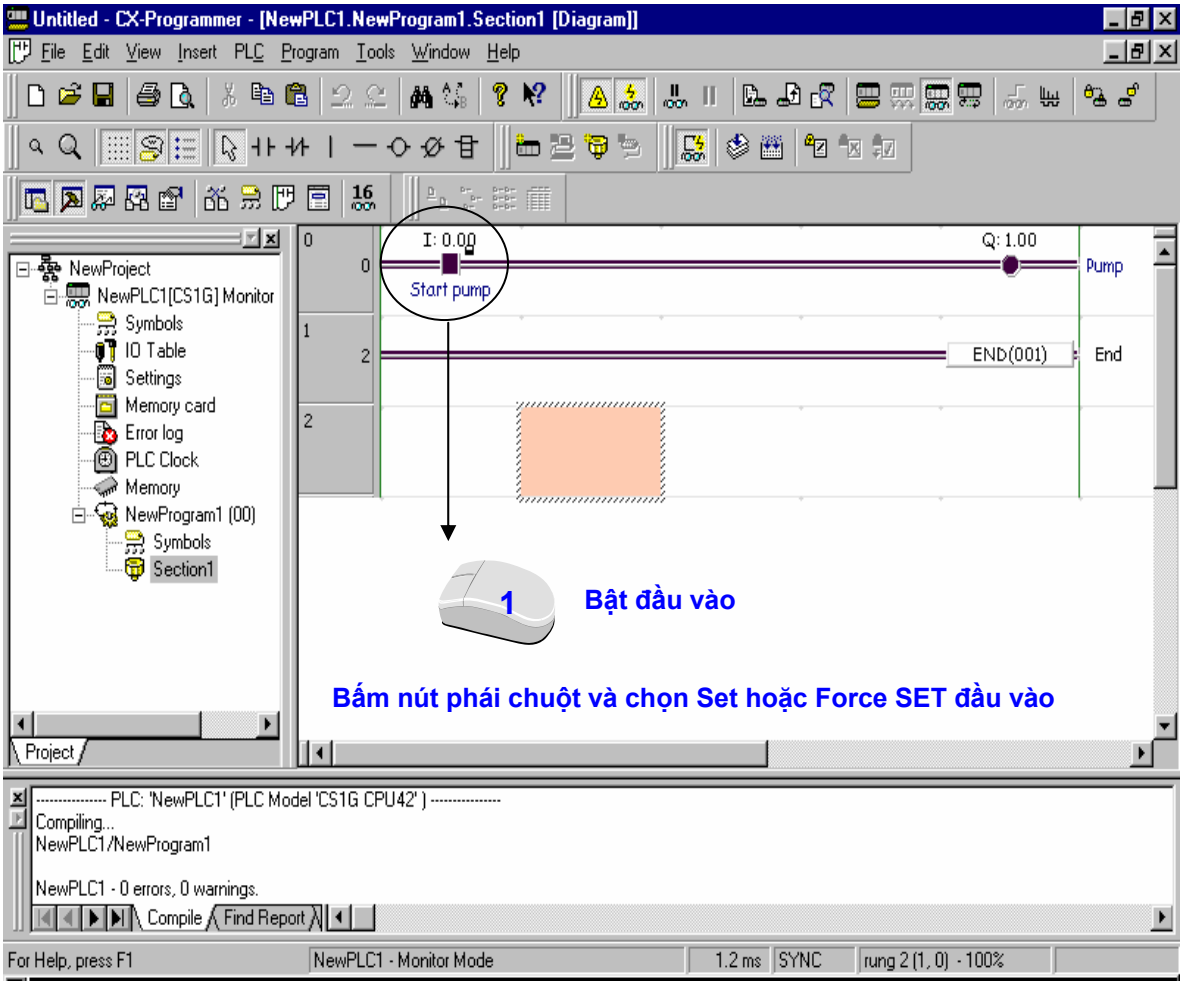
Chuyển PLC sang chế độ Monitor mode

Để chạy chương trình vừa nạp vào PLC, cần chuyển PLC sang chế độ Monitor hoặc Run mode. Ở đây ta sẽ chọn chế độ Monitor để sử dụng các chức năng khác của CX-Programmer.



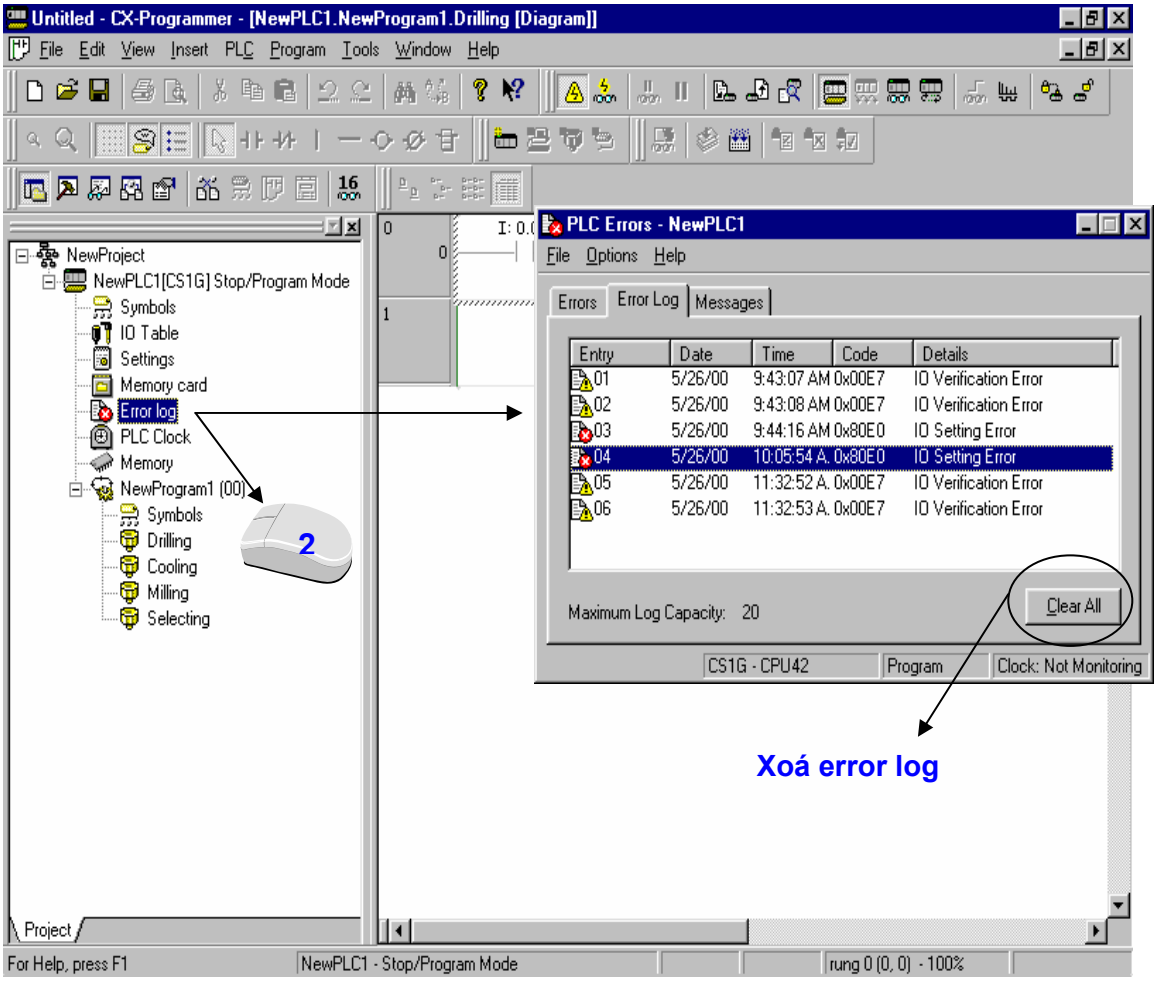
Thử chương trình

CX-Programmer có các chức năng hữu ích giúp thử và kiểm tra chương trình.
Ở đây ta có thể bật/tắt 1 bit trong chương trình hoặc đầu vào/đầu ra mà không cần đầu vào vật lý.

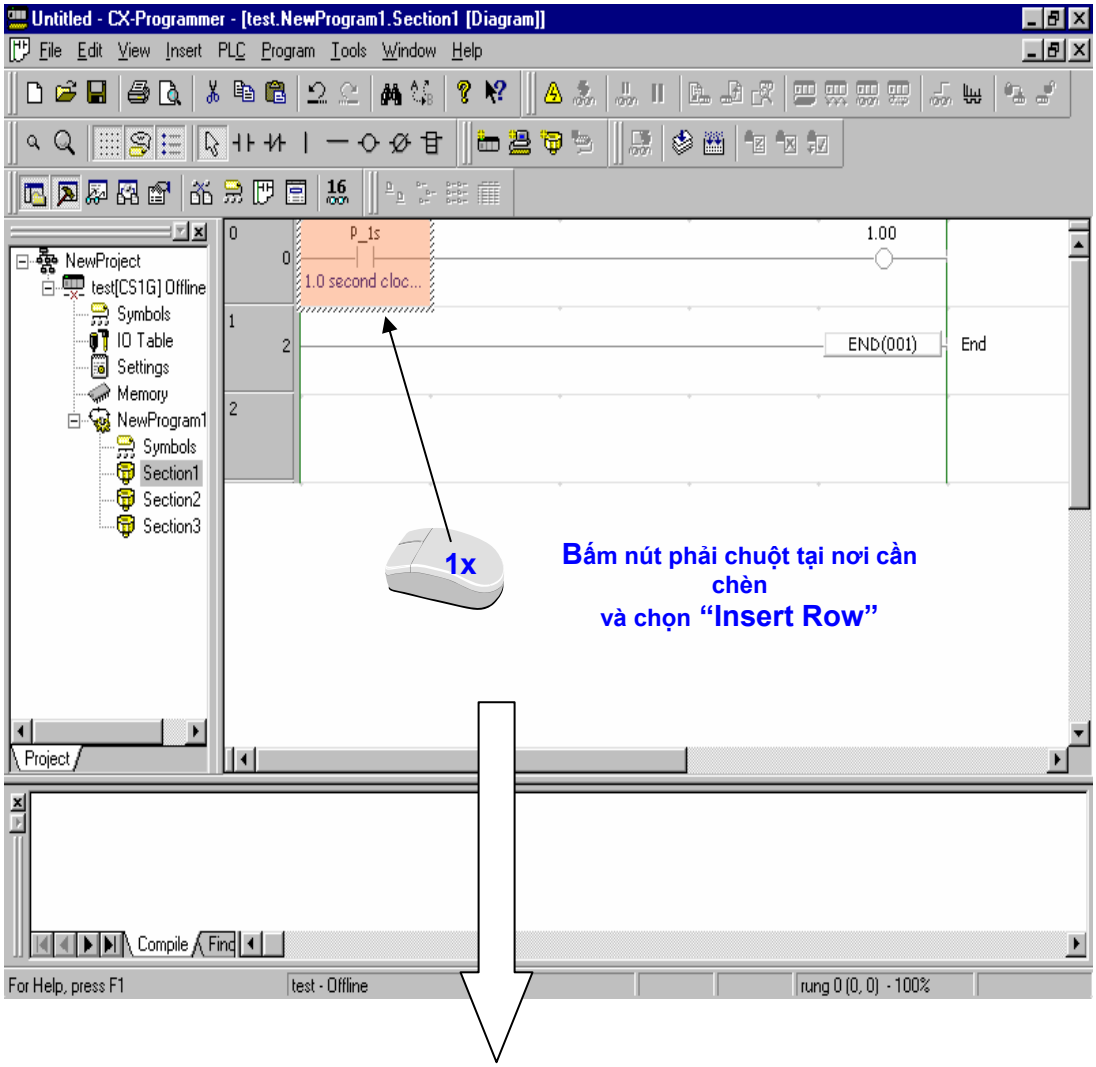


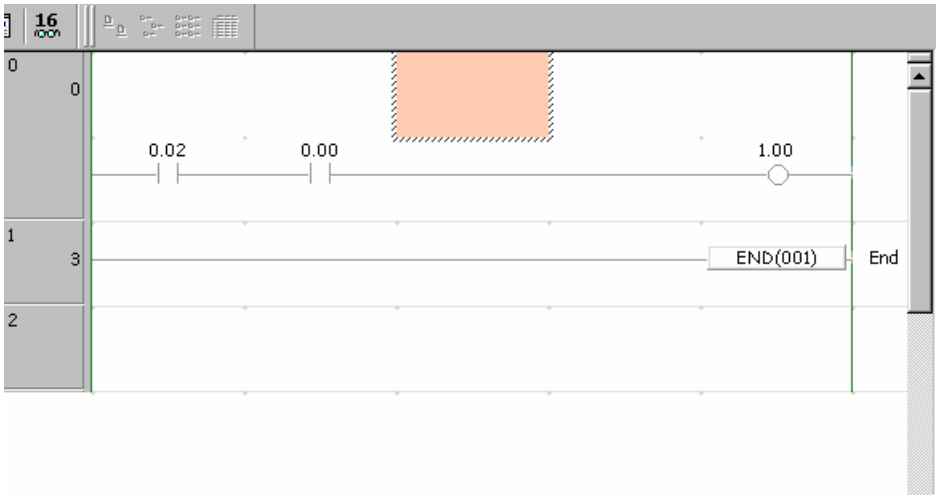
Kiểm tra bản ghi lỗi trong PLC

Khi đang online có thể kiểm tra và xóa các lỗi đang có trong PLC bằng cách bấm đúp vào Error Log.

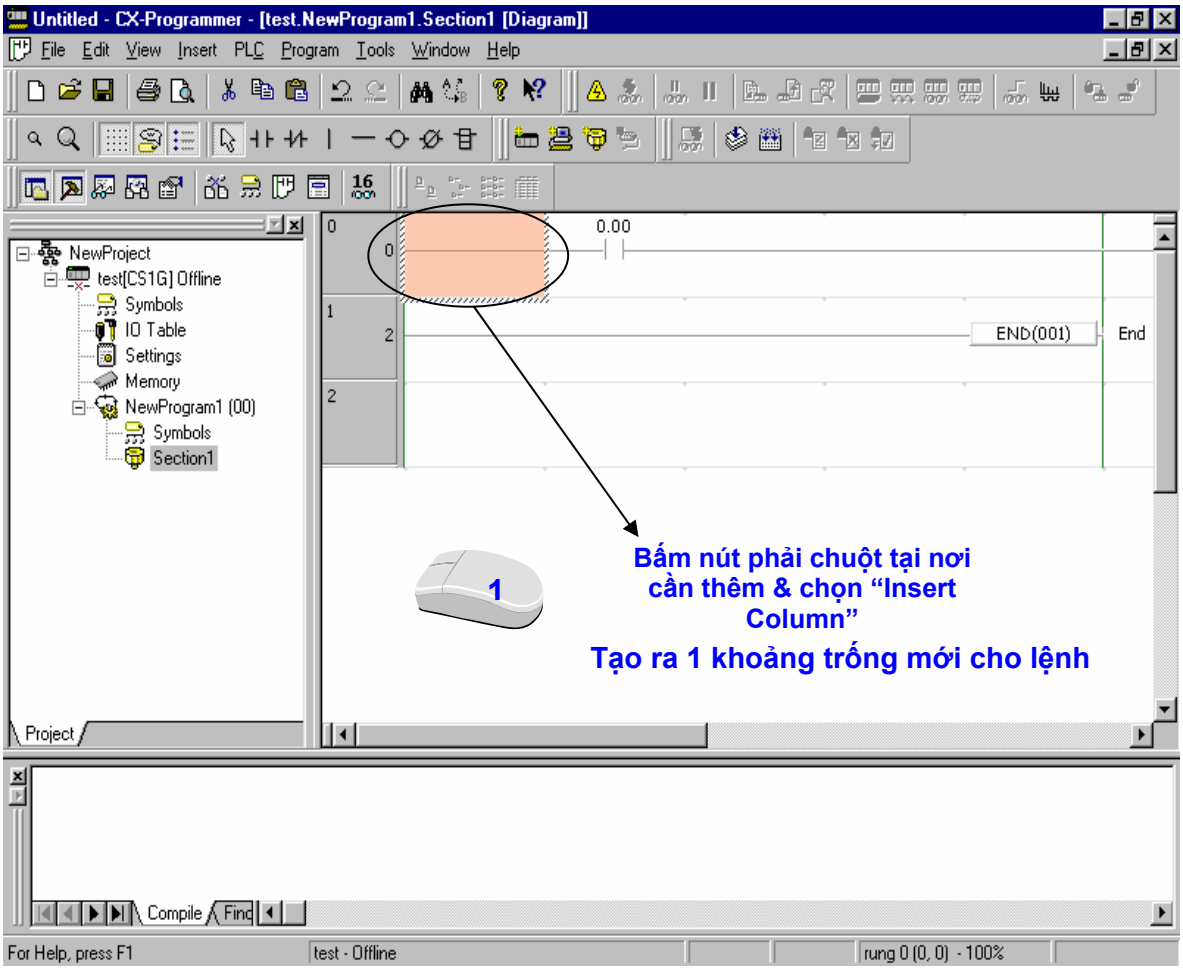


Thêm hàng vào Rung

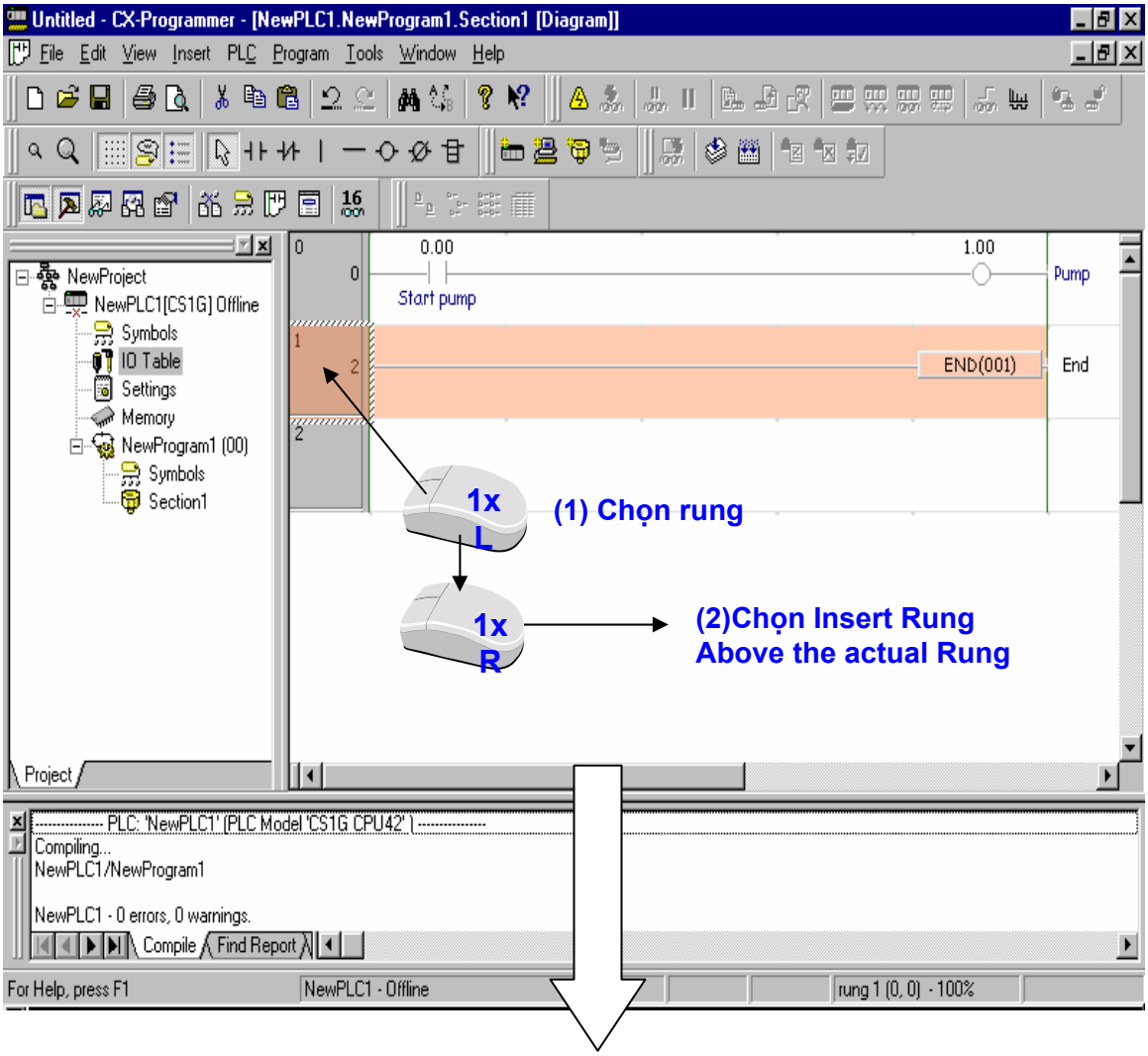


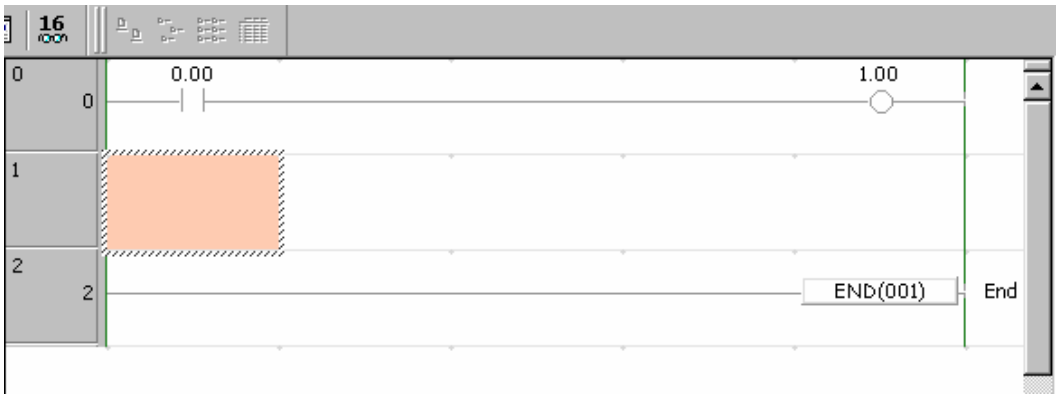


Thêm cột vào Rung



Chèn thêm 1 rung

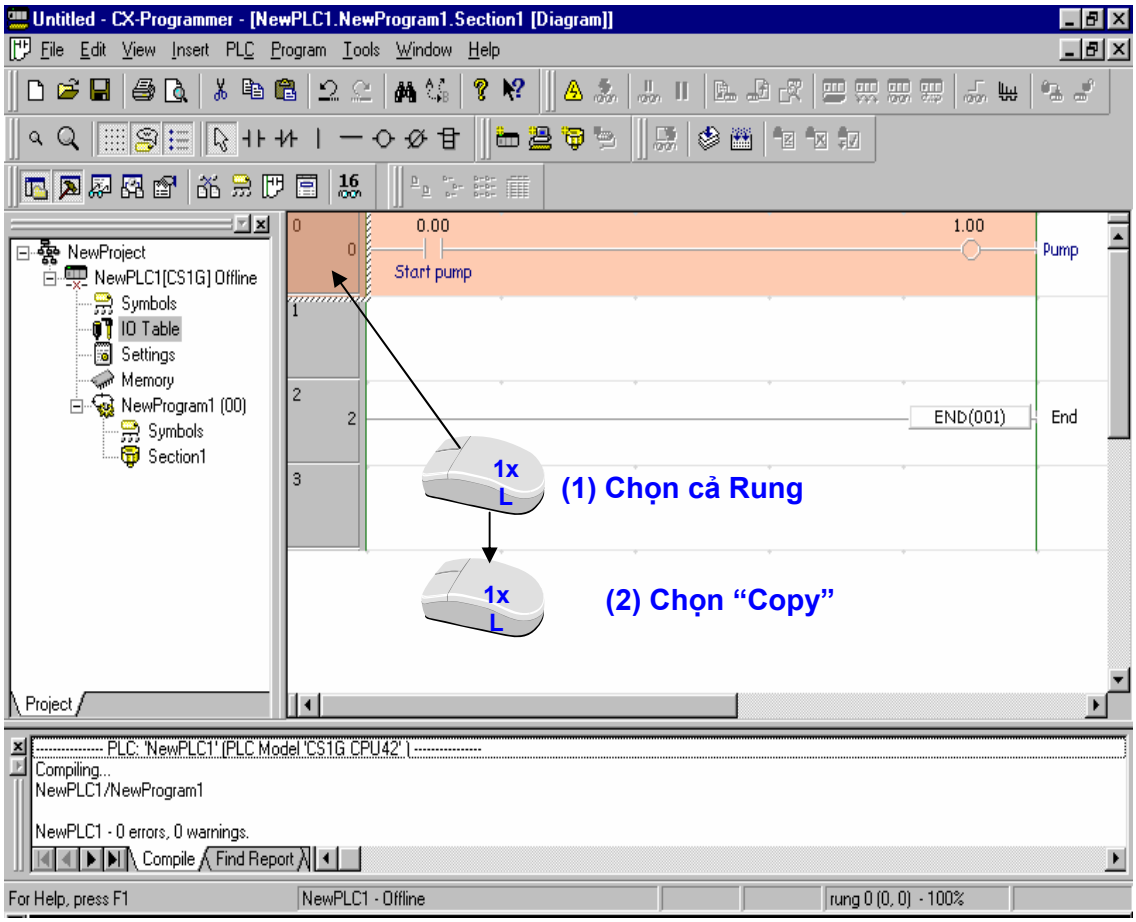




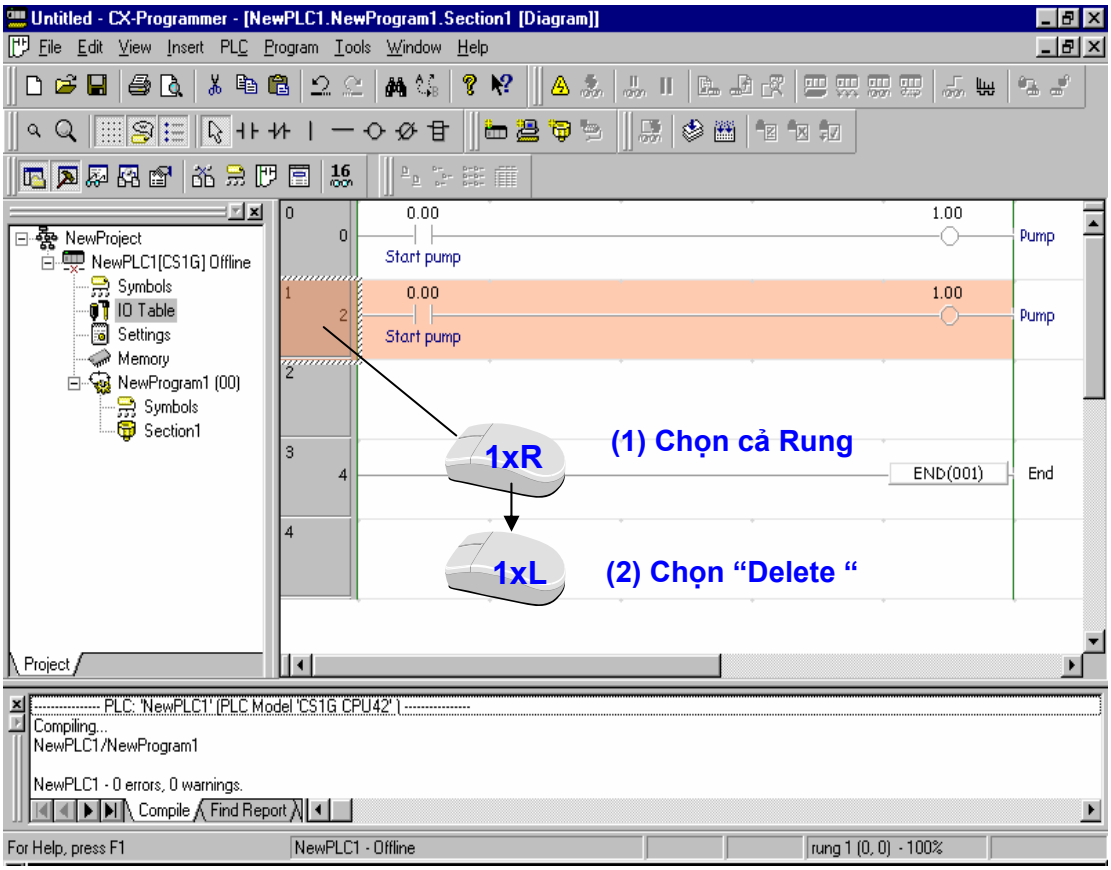
Các thao tác Copy & Paste

Ta có thể áp dụng các thao tác như Cut, Copy & Paste với các phần tử của chương trình như với 1 chương trình Windows thông thường khác. Đồng thời có thể áp dụng **Undo & Redo** với các thao tác vừa làm.

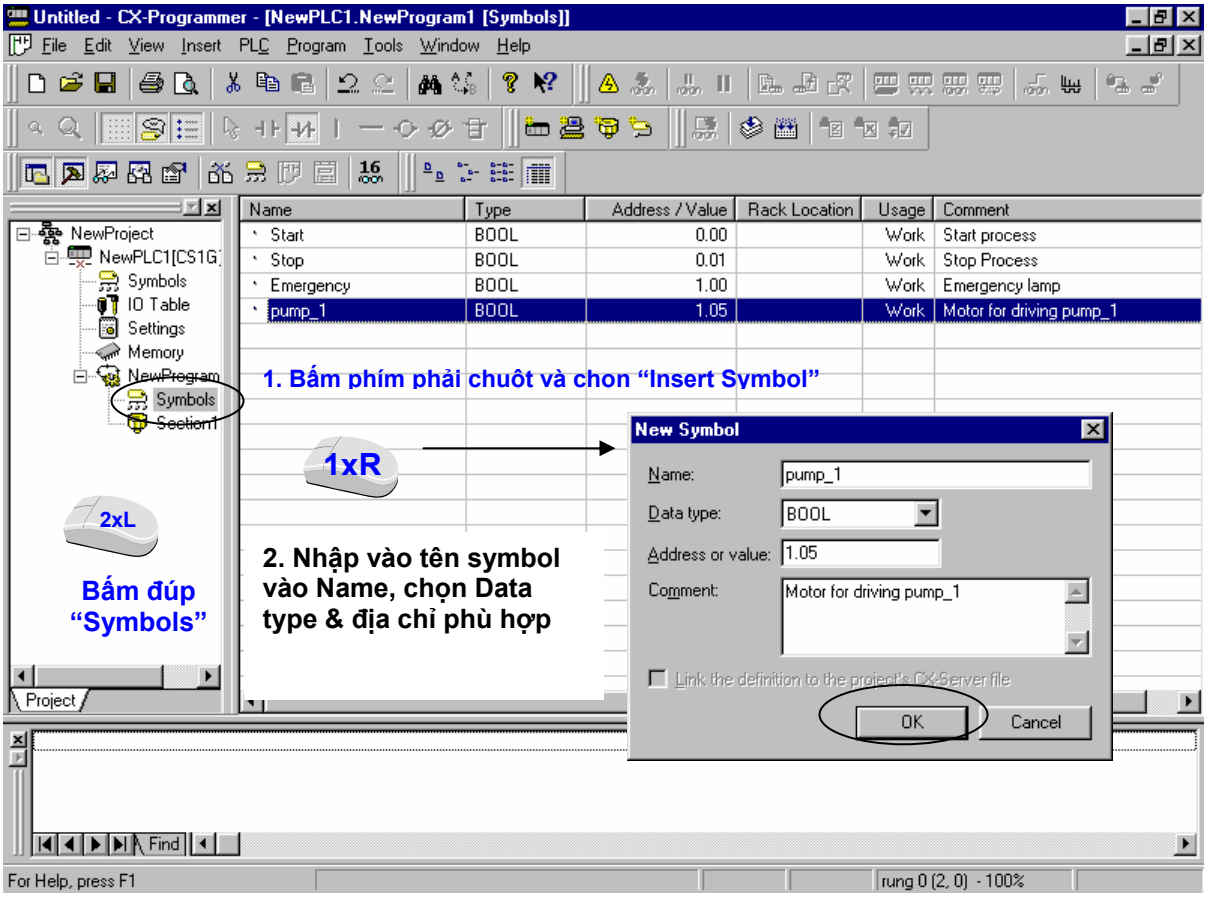
Dưới đây là ví dụ thao tác Copy cả 1 rung rồi paste vào 1 chỗ khác.



Xoá Rung

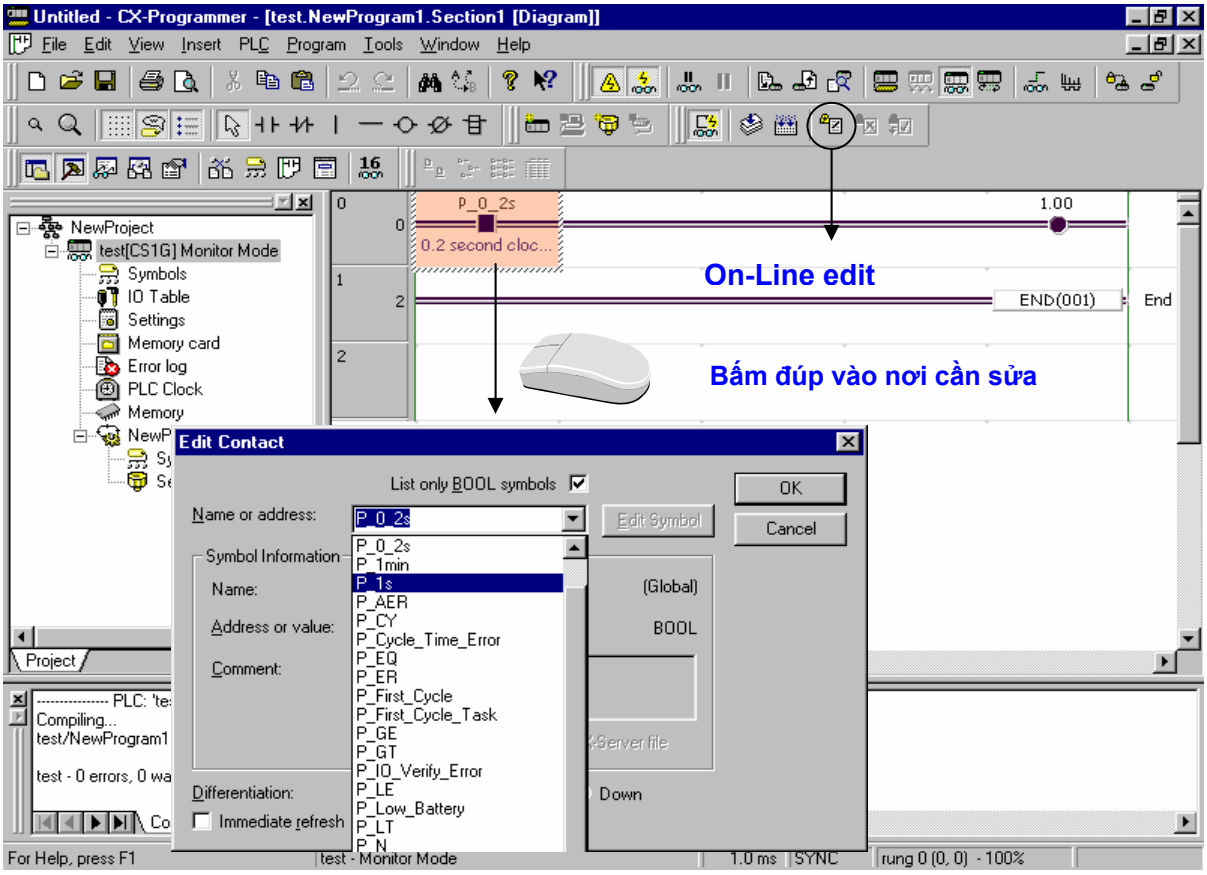

Untitled - CX-Programmer - [NewPLC1.NewProgram1.Section1 [Diagram]]
File Edit View Insert PLC Program Tools Window Help
NewProject
NewPLC1[CS1G] Offline
Symbols
IO Table
Settings
Memory
NewProgram1 (00)
Symbols
Section1
Project
0.00
1.00
Start pump
Pump
0.00
1.00
Start pump
Pump
END(001)
End
1xR
(1) Chọn cả Rung
1xL
(2) Chọn "Delete "
PLC: 'NewPLC1' (PLC Model 'CS1G CPU42')
Compiling...
NewPLC1/NewProgram1
NewPLC1 - 0 errors, 0 warnings.
Compile
Find Report
For Help, press F1
NewPLC1 - Offline
rung 1 (0, 0) - 100%

Thêm các tên (Symbol) cục bộ vào danh sách

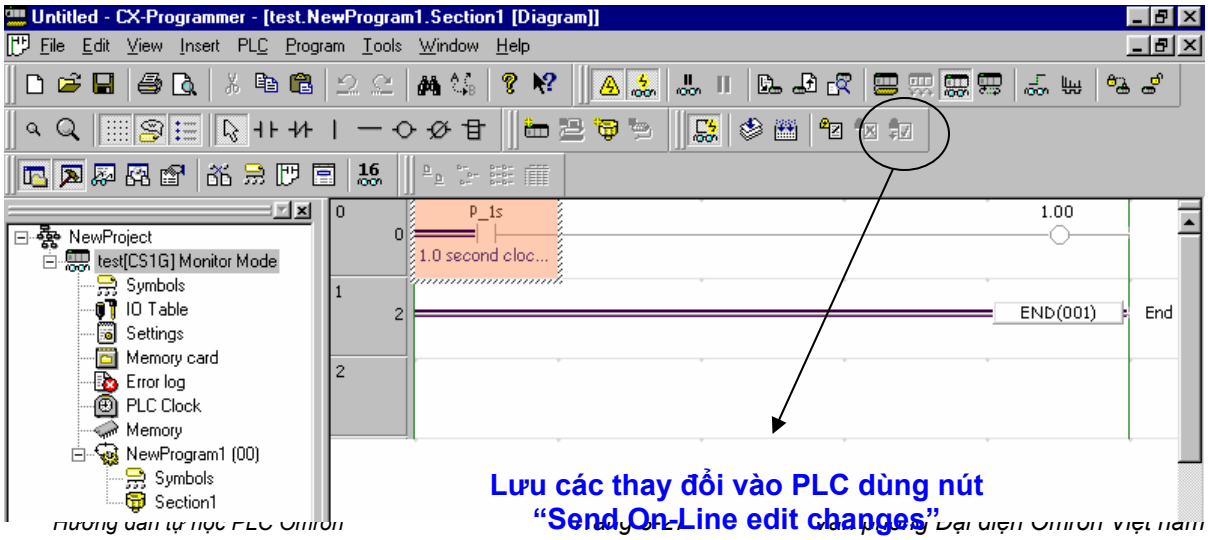


Thay đổi chương trình trực tiếp On-line

CX-Programmer cho phép sửa chương trình ngay cả khi PLC đang ở chế độ chạy bằng cách dùng tính năng **On-Line edit**.

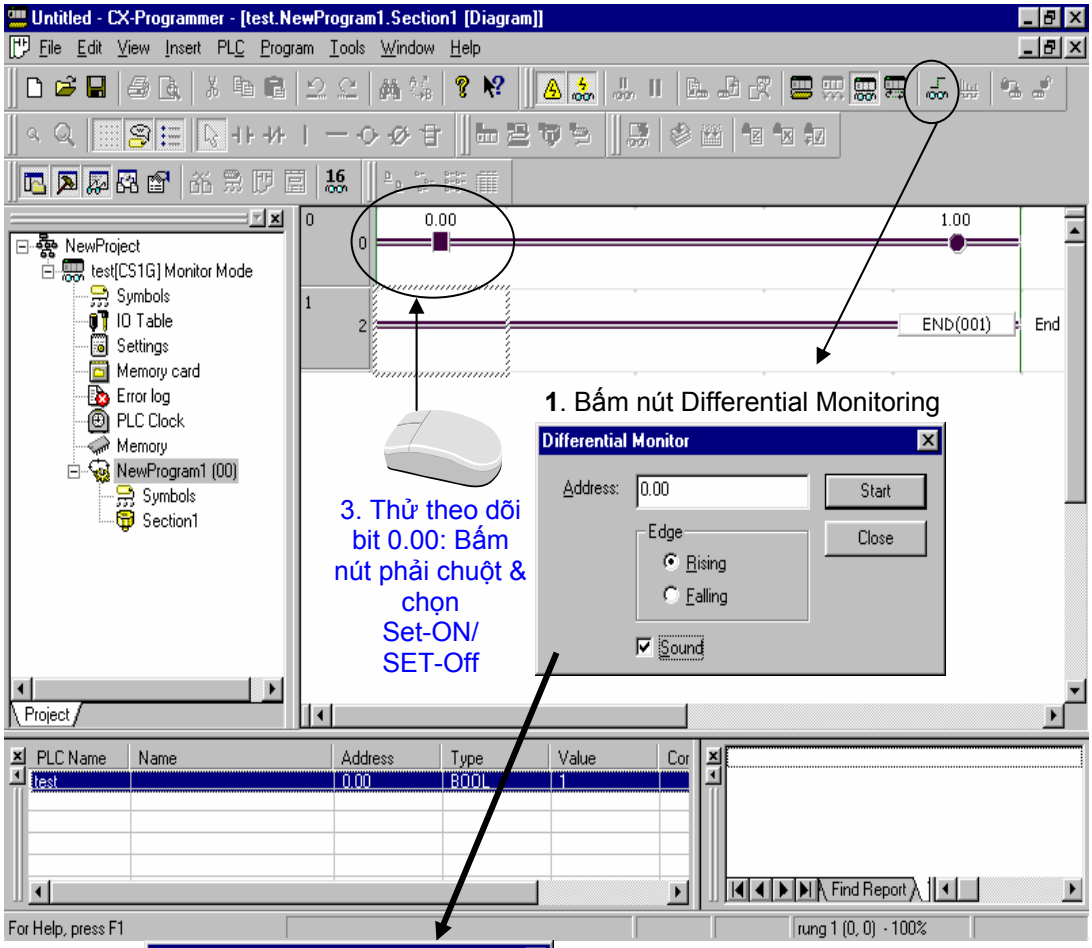


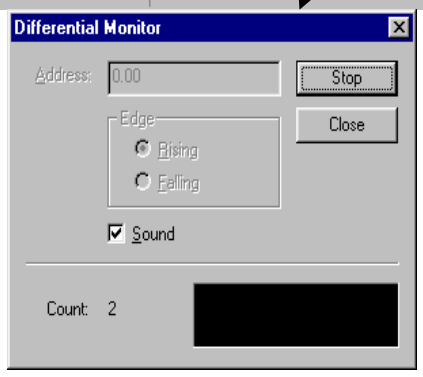
Sau khi thực hiện các thay đổi trên CX-Programmer, cần phải lưu các thay đổi này vào bộ nhớ PLC.



Theo dõi sự thay đổi (Differential Monitoring)

Với các bit thay đổi nhanh, ta có thể sử dụng tính năng này để phát hiện sự thay đổi một cách trực quan.





2. Chọn chế độ theo dõi rising/falling

4. Thay đổi màu sắc mỗi khi phần tử chuyển từ OFF → ON (Rising edge) cùng số lần chúng thay đổi

I. CẤU TRÚC VÀ CHỨC NĂNG CÁC VÙNG NHỚ

Area			Số lượng	Dài địa chỉ	Sử dụng với Task	Phân cho mục đích sử dụng	Truy cập dạng bit	Truy cập dạng Word	Cho phép		Thay đổi từ CX-Programmer	Khả năng Force Bit
									Đọc	Ghi		
CIO Area	I/O Area	Input Area	1,600 bits (100 words)	CIO 0 đến CIO 99	Được chia sẻ với tất cả các task	CP1L CPU Units & CP-series	OK	OK	OK	OK	OK	OK
		Output Area	1,600 bits (100 words)	CIO 100 đến CIO 199		Expansion Units hay Expansion I/O Units	OK	OK	OK	OK	OK	OK
	1:1 Link Area		1,024 bits (64 words)	CIO 3000 đến CIO 3063		1:1 Links	OK	OK	OK	OK	OK	OK
	Serial PLC Link Area		1,440 bits (90 words)	CIO 3100 đến CIO 3189		Serial PLC Links	OK	OK	OK	OK	OK	OK
	Work Area		14,400 bits (900 words)	CIO 3800 đến CIO 6143		---	OK	OK	OK	OK	OK	OK
Work Area			8,192 bits (512 words)	W000 đến W511		---	OK	OK	OK	OK	OK	OK
Holding Area			8,192 bits (512 words)	H000 đến H511 (Ghi chú 6)		---	OK	OK	OK	OK	OK	OK
Auxiliary Area			15,360 bits (960 words)	A000 đến A959		---	OK	---	OK	Ghi chú 1	Ghi chú 1	Không
TR Area			16 bits	TR0 đến TR15		---	OK	OK	OK	OK	Không	Không
Data Memory Area			32,768 words	D00000 đến D32767 (Ghi chú 7)		---	Không (Ghi chú 2)	OK	OK	OK	OK	Không
Timer Completion Flags			4,096 bits	T0000 đến T4095		---	OK	---	OK	OK	OK	OK
Counter Completion Flags			4,096 bits	C0000 đến C4095		---	OK	---	OK	OK	OK	OK
Timer PVs			4,096 words	T0000 đến T4095		---	---	OK	OK	OK	OK	Không (Ghi chú 4)
Counter PVs			4,096 words	C0000 đến C4095		---	---	OK	OK	OK	OK	Không (Ghi chú 5)
Task Flag Area			32 bits	TK0 đến TK31		---	OK	---	OK	Không	Không	Không
Index Registers			16 registers	IR0 đến IR15	Chức năng riêng cho từng task (Ghi chú 3)	---	OK	OK	Chỉ dùng cho đánh địa chỉ gián tiếp (Indirect addressing)	Tùy từng lệnh	Không	Không
Data Registers			16 registers	DR0 đến DR15		---	Không	OK	OK	OK	Không	Không

Ghi chú:

(1) A0 đến A447 chỉ cho phép đọc, cấm ghi. A448 đến A959 cho phép đọc/ghi (read/write)
(2) Bit này có thể được tác động bởi các lệnh TST(350), TSTN(351), SET, SETB(532), RSTB(533), & OUTB(534).
(3) Index registers & data registers có thể được dùng riêng cho từng task hay chung cho tất cả các task.
(4) Timer PVs có thể được làm tươi gián tiếp bằng cách force-setting/resetting Timer Completion Flags.
(5) Counter PVs có thể được làm tươi gián tiếp bằng cách force-setting/resetting Counter Completion Flags.
(6) H512 đến H1535 được dùng trong Function Block Holding Area. Các words có thể được dùng nội bộ bên trong các lệnh gọi function block
(7) Data Memory Area cho CPU Units với 10, 14 hay 20 I/O Points: D0 đến D9999 và D32000 đến D32767.

1- Vùng nhớ CIO (Common I/O) hay IR (Internal Relay):

Vùng nhớ Input/Output
Những bit trong vùng nhớ này dùng để đặt các địa chỉ vào/ra (I/O), nó chỉ các trạng thái ON/OFF của các tín hiệu vào/ra. Các địa chỉ không dùng cho chức năng I/O có thể sử dụng như work bit trong khi viết chương trình.

Vùng nhớ 1:1 Link Area

Vùng nhớ Serial PLC Link Area

Vùng nhớ Work bit
Các Work bit có thể được sử dụng tự do trong chương trình. Chúng chỉ sử dụng cho các mục đích bit/word trung gian trong chương trình, không thể gán cho các I/O bên ngoài.

2- Vùng nhớ Work Area:
Các bit và word trong vùng Work area có thể được sử dụng tự do trong chương trình. Chúng chỉ sử dụng cho các mục đích bit/word trung gian trong chương trình, không thể gán cho các I/O bên ngoài.

3- Vùng nhớ TR (Temporary Relay):
Sử dụng khi một sơ đồ ladder phức tạp cần phải rẽ nhánh, TR sẽ chứa tạm thời các trạng thái On/Off ở các nhánh chương trình.
TR chỉ sử dụng khi lập trình bằng mã Mnemonic. Khi lập trình bằng Ladder, TR sẽ thực hiện một cách tự động.

4- Vùng nhớ HR (Hold Relay):

Các bit HR sẽ giữ trạng thái On/Off không đổi, ngay cả khi không cấp nguồn cho PLC.

5- Vùng nhớ AR (Auxiliary Relay):
Những bit này chủ yếu phục vụ như cờ (flag), các trạng thái hoạt động của PLC.

6- Vùng nhớ Timer:
Quản lý timer được tạo ra từ các lệnh TIM, TIMH(15)
TIM dùng để truy cập cờ (nếu sử dụng bit) và giá trị hiện thời (PV) (nếu sử dụng word) của Timer

7- Vùng nhớ Counter:
Quản lý counter được tạo ra từ các lệnh CNT, CNTR(12),..
CNT dùng để truy cập cờ (nếu sử dụng bit) và giá trị hiện thời (PV) (nếu sử dụng word) của Counter.

8- Vùng nhớ DM (Data Memory):

DM chỉ có thể truy cập theo Word.
DM được chia ra hai nhóm: Nhóm sử dụng chứa các dữ liệu một cách tự do và nhóm dùng cho các chức năng đặc biệt.

9- Task Flag Area

1 cờ Task Flag sẽ lên ON khi cyclic task (task theo chu kỳ) tương ứng ở trạng thái sẵn sàng chạy (RUN) và OFF khi cyclic task chưa được thực hiện (INI) hoặc ở trạng thái chờ standby (WAIT).

10- Index Registers

Index registers (IR0 đến IR15) được dùng để lưu địa chỉ bộ nhớ PLC (địa chỉ tuyệt đối trong RAM) để đánh địa chỉ gián tiếp. Chúng có thể dùng riêng trong từng task hay chung cho tất cả các task.

11- Data registers

Data registers (DR0 đến DR15) được dùng kết hợp với Index Registers trong 1 câu lệnh để xác định địa chỉ thực cần dùng, trong đó nội dung của Data registers được cộng với nội dung của Index Registers để có địa chỉ thực. Chúng có thể dùng riêng trong từng task hay chung cho tất cả các task.

II - LỆNH CƠ BẢN KHÁC (Basic Instruction) :

Ngoài các lệnh cơ bản như:

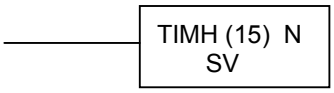
1. **Load – LD**
2. **Load Not – LD NOT**
3. **And**
4. **And Not**
5. **Or**
6. **Or Not**
7. **And Load – AND LD**
8. **Or Load – OR LD**
9. **Out, Out Not : Ngõ ra on, off khi điều kiện được thực hiện.**
10. **RESET B : Khi thực hiện bit B OFF.**
11. **SET B : Khi thực hiện bit B ON.**
12. **Keep**
13. **Timer**
14. **Counter**
15. **Reversible Counter**
16. **Differentiate Up/Down**
17. **End (01) : Điều kiện bắt buộc khi kết thúc chương trình.**

PLC OMRON còn hỗ trợ nhiều lệnh khác cho các yêu cầu điều khiển rất đa dạng trong thực tế.

Sau đây là 1 số lệnh khác.

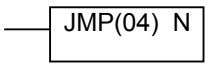
Chú ý : Số thứ tự của TIM Và CNT không được trùng nhau, tổng số TIM Và CNT là 511 đối với* CQM1, 127 *đối với CPM1, 256 đối với CPM2A.

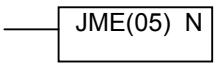
18. High-Speed Timer :



SV : 0.01 đến 99.99 Sec

19. Jump – Jump End :





Jump 00 :

Nếu N=00, CPU sẽ tìm JME(05) kế đó với số thứ tự tương ứng là 00. Khi thực hiện việc tìm kiếm này, chu kỳ quét của chương trình sẽ dài hơn so với thực hiện các Jump với số thứ tự N # 0.

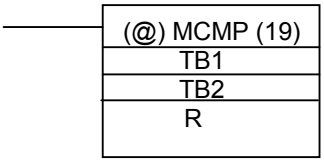
Trạng thái của Timer, Counter, Output bit, và tất cả các trạng thái khác của những lệnh ở giữa JMP(04) và JME(05) sẽ không thay đổi. Jump có số thứ tự 00 có thể dùng nhiều lần trong một chương trình.

- *DIFU(13) và DIFD(14) trong lệnh Jump:*
  Giả sử một bit ON bởi DIFU(13) hay DIFD(14) đặt trong JMP(04) và JME(05), đến chu kỳ quét kế tiếp, nếu điều kiện của Jump là OFF, bit đó sẽ vẫn giữ trạng thái ON cho đến khi điều kiện của Jump bật lên ON (tức là khi chương trình không thực hiện rẽ nhánh).

Chú ý: Khi JMP(04) và JME(05) không sử dụng theo từng cặp, một thông báo lỗi (Error) sẽ hiển thị *khi thực hiện việc kiểm tra chương trình, tuy nhiên chương trình vẫn hoạt động bình thường.*

III- LỆNH SO SÁNH DỮ LIỆU :

1. Multi-Word compare

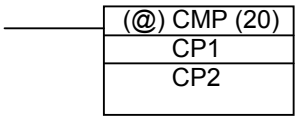


(CPM1 không có lệnh này).
TB1 đến TB1+15 phải trên cùng vùng dữ liệu.
TB2 đến TB2+15 phải trên cùng vùng dữ liệu.

MCMP(19) so sánh nội dung của TB1 và TB2, TB1+1 và TB2+2, ..., TB1+15 và TB2+15. Kết quả của bit tương ứng trên word R sẽ Off nếu hai word so sánh tương ứng bằng nhau .

- Bit P_EQ On nếu nội dung cả hai TB1 và TB2 bằng nhau :R = 0000.
- Bit P_ER On nếu xảy ra lỗi khi thực hiện lệnh .

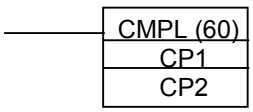
2. Compare :



So sánh nội dung của hai word CP1 và CP2 :

- CP1 < CP2 : LE (P_LE On)
- CP1 = CP2 : EQ (P_EQ On)
- CP1 > CP2 : P_GT (P_GT On)

3. Double Compare :

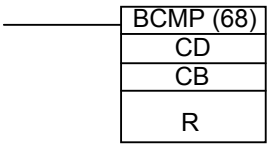


CP1 và CP1+1 phải trên cùng vùng dữ liệu.
CP2 và CP2+1 phải trên cùng vùng dữ liệu.

CMPL (60) nối 4 digit của CP1+1 và CP1 thành một 2 word (8 digits) , Cp2+1 và CP2 thành hai word (8 digits) , trong đó những bit của CP1+1 và CP2+1 là những bit có trọng số lớn nhất. Tiếp theo CMPL (60) sẽ so sánh nội dung của hai word này :

- CP1+1,CP1 < CP2+1,CP2 : LE (P_LE On)
- CP1+1,CP1 = CP2+1,CP2 : P_EQ (P_EQ On)
- CP1+1,CP1 > CP2+1,CP2 : P_GT (P_GT On)

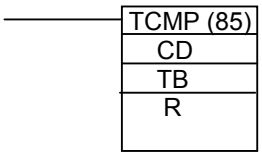
4. Block Compare :



CD : Dữ liệu được so sánh
CB : Word đầu tiên của khối word (block word) cần so sánh .(Nội dung của word giới hạn nhỏ nhất phải nhỏ hơn hay bằng nội dung của word giới hạn lớn nhất)
R : Word trả về kết quả. (DM 6144 đến DM 6655 không được sử dụng).

BCMP (68) thực hiện so sánh CD với những khoảng được tạo ra trong khối word bắt đầu từ CB. Kết quả trả về bit tương ứng trong R.
CB<=CD<=CB+1 : Bit 00
CB+2<=CD<=CB+3 : Bit 01
CB+3<=CD<=CB+4 : Bit 02
.....
CB+30<=CD<=CB+31: Bit 15

5. Table Compare :

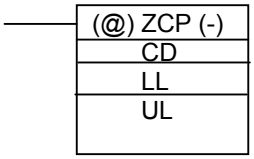


CD : Dữ liệu được so sánh
TB : Word đầu tiên trong bảng cần so sánh
R : Word trả về kết quả

TCMP (85) thực hiện so sánh CD với nội dung của các word trong bảng TB, TB+1,..., TB+15. Nếu CD bằng bất kỳ nội dung của word nào trong bảng thì bit tương ứng của R sẽ On .

P_ER : Bit P_ER sẽ On nếu điều kiện thực hiện lệnh không đúng.

6. Area Range Compare :



Lệnh này chỉ thực hiện cho PLC loại CJ1M, CJ1, CS1. Các cờ LE, P_EQ & P_GT là các cờ đặc biệt ở vùng nhớ riêng.

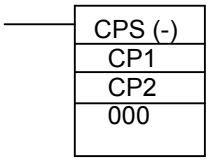
CD : Dữ liệu được so sánh
LL : Giới hạn dưới

UL : Giới hạn trên.
LL : Phải nhỏ hơn hay bằng UL

- CD < LL : LE (P_LE On)
- LL <=CD <=UL : P_EQ (P_EQ On)
- UL<CD : P_GT (P_GT On)

P_ER : Bit P_ER sẽ On nếu điều kiện thực hiện lệnh không đúng.

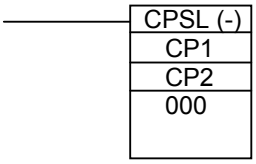
7. Signed Binary Compare :



Lệnh này chỉ thực hiện cho PLC loại CJ1M, CJ1, CS1. Các cờ LE, P_EQ & P_GT là các cờ đặc biệt ở vùng nhớ riêng

CP1 : Word so sánh thứ nhất
CP2 : Word so sánh thứ hai
000 : Không sử dụng.
CPS (-) so sánh 16 bit nhị phân có dấu (giá trị đại số) của CP1 và CP2 , kết quả trả về bit P_LE, P_EQ, P_GT.

8. Double Signed binary Compare :

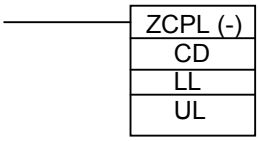


Lệnh này chỉ thực hiện cho PLC loại CJ1M. Các cờ LE, P_EQ & P_GT là các cờ đặc biệt ở vùng nhớ riêng

CP1 : Word so sánh thứ nhất
CP2 : Word so sánh thứ hai
000 : Không sử dụng

CPSL (-) so sánh 32 bit có dấu của CP1+1,CP1 cà CP2+1,CP2 . Kết quả trả về bit P_LE, P_EQ, P_GT.

9. Double Area Range Compare :

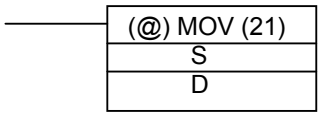


Lệnh này chỉ thực hiện cho PLC loại CJ1M, CP1L/1H. Các cờ LE, P_EQ & P_GT.

CD : Dữ liệu được so sánh
LL : Giới hạn dưới của khoảng cần so sánh
UL : Giới hạn trên của khoảng cần so sánh

IV- LỆNH TRUYỀN DỮ LIỆU :

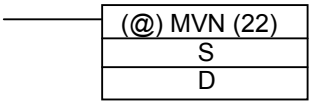
1. Move :



MOV(21) copy nội dung của S vào D.

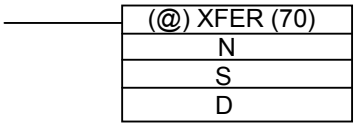
P_ER : Bit P_ER sẽ On nếu điều kiện thực hiện lệnh không đúng.
P_EQ : Bit P_EQ On khi nội dung copy vào D là 0

2. Move Not :



MVN(22) copy phủ định của nội dung của S sang D.
P_ER : Bit P_ER sẽ On nếu điều kiện thực hiện lệnh không đúng.
P_EQ : Bit P_EQ On khi nội dung copy vào D là 0

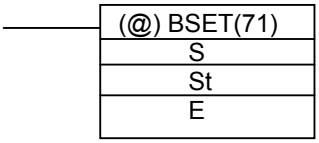
3. Block Transfer :



S đến S+N phải trên cùng vùng dữ liệu
D đến D+N phải trên cùng vùng dữ liệu
N : Số Word cần truyền (BCD)

XFER (70) copy nội dung các word S,..., S+N sang D,..., D+N theo thứ tự.
P_ER : Bit P_ER sẽ On nếu điều kiện thực hiện lệnh không đúng.

4. Block Set :

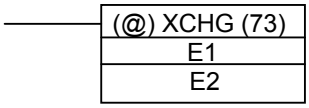


S : Dữ liệu nguồn
St : Word bắt đầu
E : Word kết thúc

St phải nhỏ hơn hay bằng E, St và E phải trên cùng một vùng dữ liệu.

BSET (71) copy nội dung của S đến tất cả các word từ St đến E.
BSET (71) có thể dùng để thay đổi giá trị đặt (PV) của timer/counter (điều này không thể thực hiện bởi MOV, MVN). BSET (71) còn được sử dụng để xoá một vùng dữ liệu, vùng DM bằng cách copy 0 đến tất cả các word của vùng dữ liệu muốn xoá.

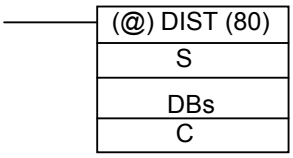
5. Data Exchange :



XCHG (73) thực hiện chuyển đổi nội dung giữa hai word E1 và E2.
Có thể chuyển đổi nội dung của block word bằng cách kết hợp với lệnh XFER.

P_ER : Bit P_ER sẽ On nếu điều kiện thực hiện lệnh không đúng.

6. Single Word Distribute :

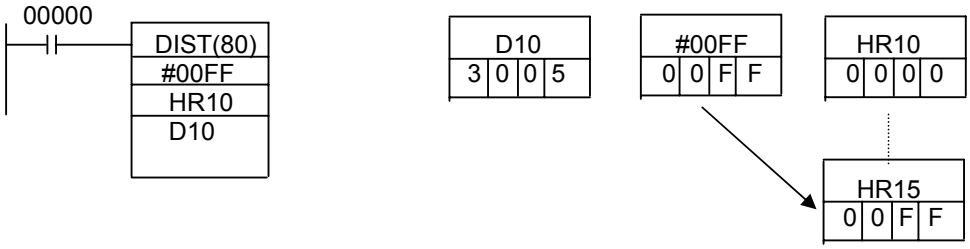


S : Word nguồn.
DBs : Word cần copy
C: Word điều khiển (BCD). (bit 00 đến 11 là offset – Off)

Nếu bit 12 đến 15 của C bằng 0 ~ 8 (BCD), DIST(80) sẽ copy nội dung của S vào DBs+Of, trong đó nội dung của C chính là Offset.

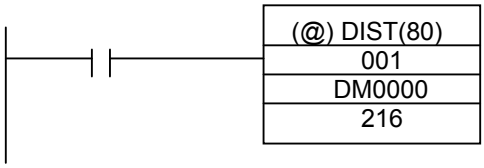
Chú ý: DBs và DBs+Of phải trên cùng vùng dữ liệu

Td: DIST(80) copy #00FF vào HR+Of. Nội dung của D10 là #3005, vì thế #00FF được copy vào HR15 (HR10+5) khi IR00000 On.



Nếu bit 12 đến 15 của C bằng 9 (BCD), DIST(80) thực hiện thao tác copy S vào ngăn xếp từ DBs+1 đến DBs+Of được trỏ đến.
DBs : Con trỏ ngăn xếp.
Bit 00 đến 11 của C: Xác định số lượng ngăn xếp. (000 ~999).

Td: DIST(80) tạo một ngăn xếp từ DM0001 đến DM0005. DM0000 tác động như một con trỏ ngăn xếp.

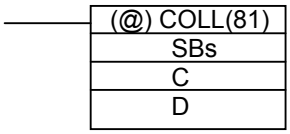


IR001	FFFF
IR216	9005

DM0000	0000		DM0000	0001		DM0000	0002	
DM0001	0000	First execution	DM0001	FFFF	Second execution	DM0001	FFFF	
DM0002	0000	→	DM0002	0000	→	DM0002	FFFF	
DM0003	0000	Stact pointer	DM0003	0000	Stact pointer	DM0003	0000	
DM0004	0000	incremented	DM0004	0000	incremented	DM0004	0000	
DM0005	0000		DM0005	0000		DM0005	0000	

P_ER : Bit P_ER sẽ On nếu điều kiện thực hiện lệnh không đúng.
P_EQ : On khi nội dung của S là 0.

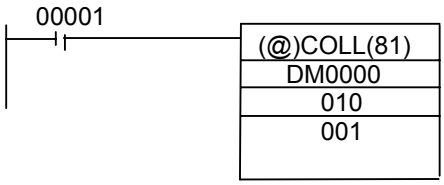
7. Data Collect :



SBs : Word nguồn.
C : Word điều khiển (Bit 00 đến 11 là offset - Of). C phải là BCD.
D : Word cần copy.

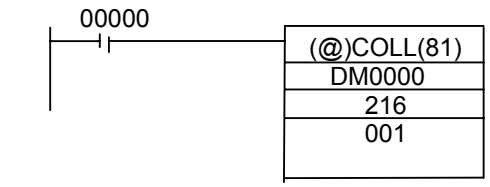
- Nếu bit 12 đến 15 của C = 0 đến 7, nội dung của C là Offset (Of). COLL(81) copy nội dung của SBs+Of vào D. SBs và SBs+Of phải trên cùng vùng dữ liệu.

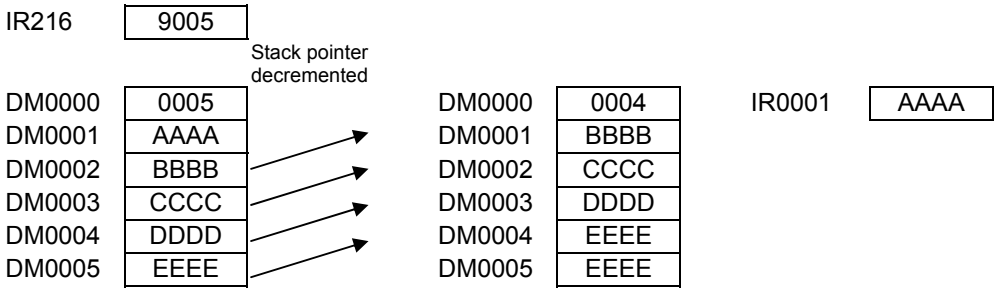
Td: COLL(81) copy nội dung của DM0000+Of vào IR001. Nội dung của 010 là #0005, vì thế nội dung của DM0005 (DM0000+5) được copy vào IR001 khi điều kiện IR00001 On.



- Nếu bit 12 đến 15 của C bằng 9, COLL(81) thực hiện thao tác FIFO ngăn xếp , trong đó D là word mà nội dung sẽ trả về nội dung của ngăn xếp được trỏ tới, SBs là con trỏ ngăn xếp, bit 00 đến 11 của C xác định số lượng ngăn xếp.

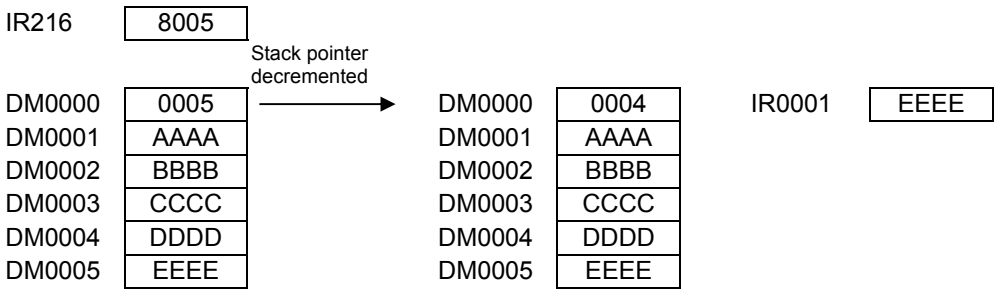
Td: Tạo một ngăn xếp giữa DM0001 và DM0005, DM0000 là con trỏ ngăn xếp.
Khi IR00000 chuyển từ Off sang On, COLL(81) dịch dữ liệu của DM0002 đến DM0005 lên một một địa chỉ, dữ liệu của DM0000 dịch sang IR001, nội dung của con trỏ ngăn xếp giảm đi 1.





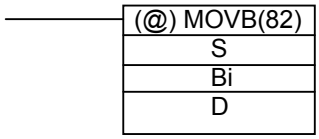
- Nếu bit 12 đến 15 của C bằng 8, COLL(81) thực hiện thao tác LIFO ngăn xếp, trong đó D là word mà nội dung sẽ trả về nội dung của ngăn xếp được trỏ tới, SBs là con trỏ ngăn xếp, bit 00 đến bit 11 của C xác định số lượng ngăn xếp.

Td: COLL(81) tạo một ngăn xếp giữa DM0001 và DM0005. DM0000 là con trỏ ngăn xếp.
Khi IR 00000 chuyển từ Off sang On, COLL(81) copy nội dung của DM0005 (DM0000+5) vào IR001. Nội dung của con trỏ ngăn xếp giảm một đơn vị.



P_ER : Bit P_ER sẽ On nếu điều kiện thực hiện lệnh không đúng.
P_EQ : On khi nội dung của SBs là 0.

8. Move Bit :



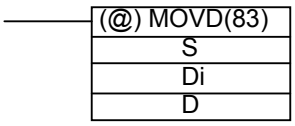
S : Word nguồn.
D : Word cần copy.
Bi : Word chỉ định.(BCD)
Hai số có trọng số nhỏ nhất (LSB) chỉ định số thứ tự của bit cần copy trong S. (00 đến 15)

Hai số có trọng số lớn nhất (MSB) chỉ định số thứ tự của bit được copy trong D.(00 đến 15)

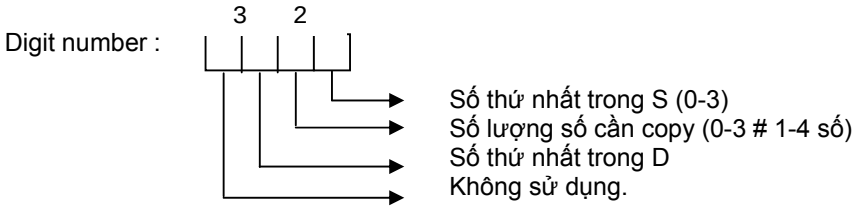
MOVB(82) sẽ copy bit được chỉ định trong S đến bit được chỉ định trong D.

P_ER : Điều kiện thực hiện lệnh không đúng.

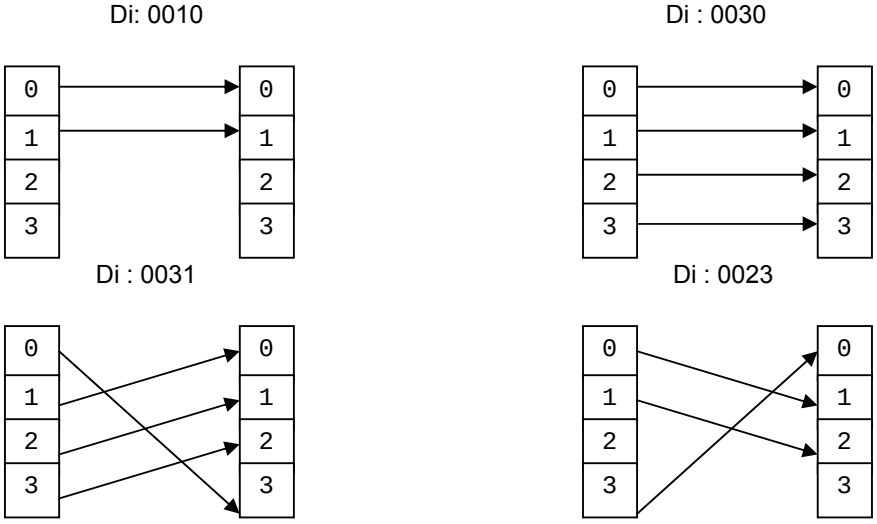
9. Move Digit :



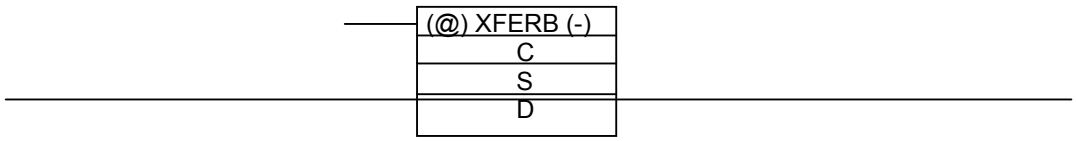
S : Word nguồn.
D : Word cần copy.
Di: Word chỉ định (BCD) .



MOVD(83) copy lần lượt những bit trong S được xác định bởi Di đến những bit trong D cũng được xác định bởi Di.

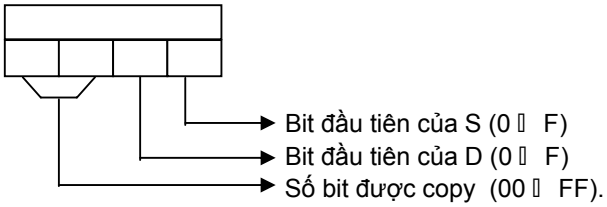


10. Transfer Bits :



Lệnh này chỉ sử dụng cho PLC loại CJ1M, CP1L/1H, CJ1 & CS1.

S : Word nguồn đầu tiên.
D : Word cần copy đầu tiên.
C : Word control.
Hai số bên phải của C đặc trưng cho bit bắt đầu trong S và D, hai bit bên trái xác định số bit được copy.



(Có thể copy đến 255 bits).
XFRB(-) copy những bit của word nguồn đến những bit của word cần copy được xác định bởi C.

P_ER : Bit P_ER sẽ On nếu điều kiện thực hiện lệnh không đúng.

V- LỆNH CHUYỂN ĐỔI :

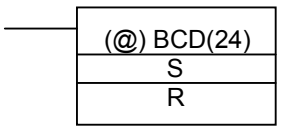
1. BCD-To-Binary :

S : Word nguồn ; R : Word kết quả.

BIN(23) chuyển đổi nội dung BCD của S sang dạng nhị phân (Binary), kết quả trả về Word R. Sau khi lệnh thực hiện , nội dung của R bị thay đổi, nội dung của S vẫn giữ nguyên.

P_ER : On khi nội dung của S không phải dạng BCD hay địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả là 0.

2. Binary-To-BCD :



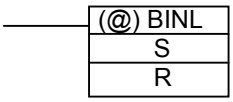
S : Word nguồn ; R : Word kết quả.

Nếu S chứa giá trị lớn hơn 270F, kết quả chuyển đổi sẽ lớn hơn 9999 nên BCD(24) không thể thực hiện được. Khi lệnh không thực hiện được, nội dung trong R vẫn không thay đổi.

BCD (24) chuyển đổi giá trị dạng nhị phân (hexadecimal) của S sang giá trị tương đương dạng BCD, kết quả trả về Word R. Sau khi thực hiện lệnh, chỉ nội dung trong R bị thay đổi, nội dung trong S không thay đổi.

P_ER : On khi địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả là 0.

2. Double BCD-To-Double Binary :

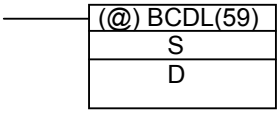


Lệnh này chỉ sử dụng cho PLC loại CJ1M, CP1L/1H, CJ1 & CS1.

Nội dung trong S phải dưới dạng BCD.
BINL(58) chuyển đổi nội dung dạng BCD của S, S+1 sang dạng nhị phân 32 bits, kết quả trả về R.

P_ER : On khi nội dung trong S, S+1 không phải dạng BCD hoặc khi địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả là 0.

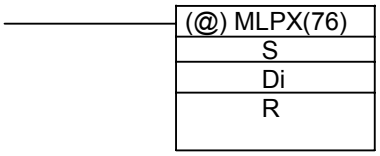
3. Double Binary-To-Double BCD :



Lệnh này chỉ sử dụng cho PLC loại CJ1M, CP1L/1H, CJ1 & CS1.
Nội dung của S phải dưới dạng nhị phân (Binary).
Nếu nội dung của S lớn hơn 05F5E0FF,khi đó kết quả chuyển đổi sẽ lớn hơn 99999999 , BINL(59) sẽ không được thực hiện. Khi lệnh không được thực hiện, nội dung trong R, R+1 vẫn không thay đổi.

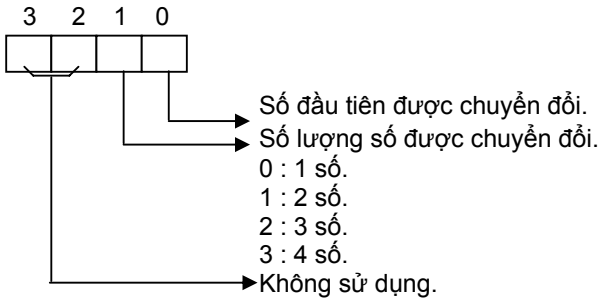
BCDL(59) chuyển đổi nội dung dạng nhị phân 32 bits của S sang dạng BCD, kết quả trả về R.
P_ER : On khi nội dung của R, R+1 vượt quá 99999999, hoặc khi địa chỉ gián tiếp DM không tồn tại.
P_EQ : On khi kết quả là 0.

4. 4-To-16 Decoder :



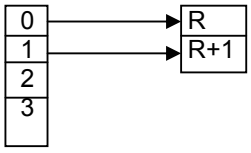
S : Word nguồn.
Di: Word chỉ định. Hai số bên phải của Di phải nằm trong khoảng 0-3.
R : Word kết quả đầu tiên. (Tất cả Word kết quả phải trên cùng vùng dữ liệu).

MLPX(76) chuyển đổi 4 số thập lục phân (hexadecimal) trong S sang giá trị thập phân từ 0 đến 15, tương ứng với mỗi giá trị thập phân đó sẽ xác định vị trí chuyển sang On trong R. Word Di sẽ chỉ định vị trí số đầu tiên và số lượng số được chuyển đổi trong S.

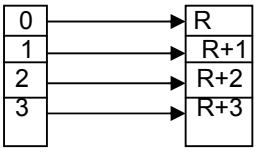


Ví dụ :

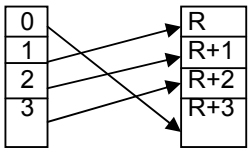
Di: 0010



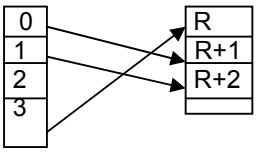
Di: 0030



Di: 0031

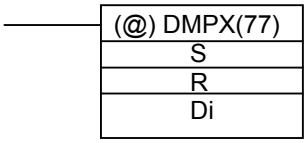


Di: 0023



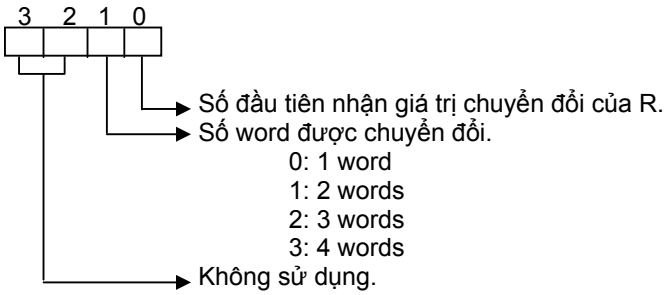
P_ER : Nếu nội dung của Di không thoả, hay số lượng R, R+1,..., vượt ngoài vùng dữ liệu.
Địa chỉ gián tiếp của DM không tồn tại.

5. 16-To-4 Encoder :



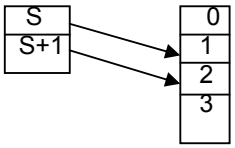
S : Word nguồn thứ nhất.
R : Word kết quả.
Di : Word chỉ định.
Hai số bên phải của Di phải nằm trong khoảng 0-3.
Tất cả word nguồn phải trên cùng vùng dữ liệu.
DMPX(77) xác định bit ở trạng thái On có trọng số lớn nhất trong S, mã hoá vị trí bit đó sang dạng thập lục phân, sau đó truyền giá trị này vào R được xác định bởi Di.

Nội dung trong Di được định nghĩa như sau :

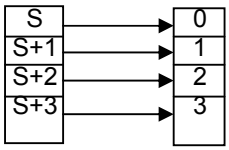


Ví dụ :

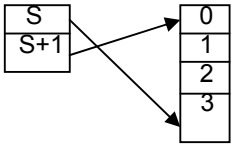
Di: 0011



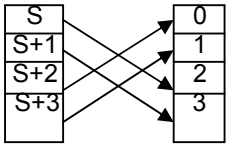
Di: 0030



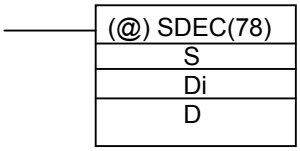
Di: 0013



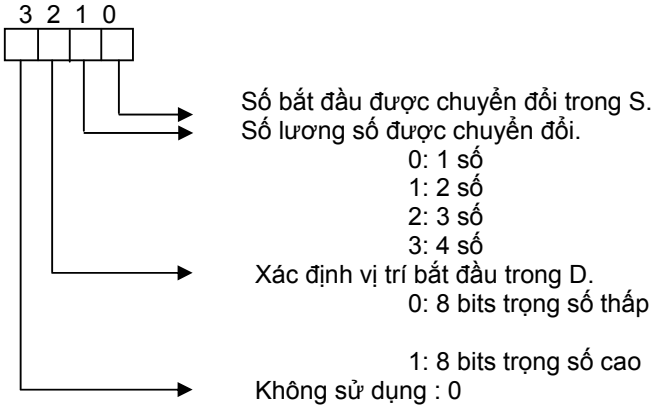
Di: 0032



6. 7-Segment Decoder :



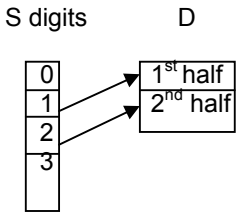
S : Word nguồn.
D : Word đích đầu tiên . (Tất cả các word D, D+1,..., phải trên cùng một vùng dữ liệu).
Di : Word chỉ định .



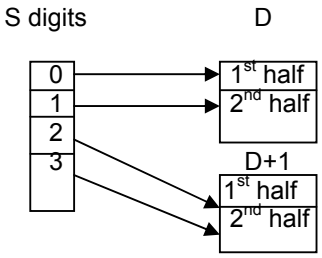
SDEC(78) chuyển đổi mỗi 4 bits trong S (được xác định bởi Di) sang mã 7 đoạn , kết quả trả về D.

Ví dụ:

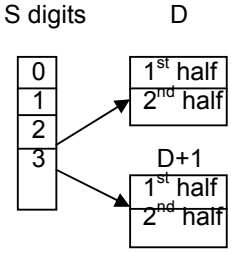
Di: 0011



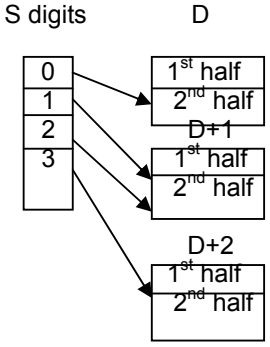
Di: 0030



Di: 0112



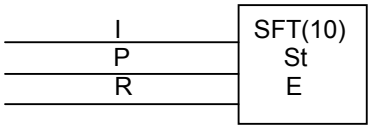
Di: 0130



P_ER : Khi những số trong word chỉ định không đúng, hoặc D vượt ngoài vùng dữ liệu.
Địa chỉ gián tiếp của DM không tồn tại.

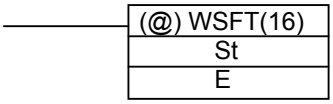
V- LỆNH DỊCH DỮ LIỆU (Shift Instruction) :

1. Shift Register :



St : Word bắt đầu.
E : Word kết thúc.
E phải lớn hơn hay bằng St, E và St phải trên cùng một Word.
I : Điều kiện thực hiện lệnh.
P : Xung tác động.
R : Reset.
Khi xung P thay đổi từ Off sang On (tác động cạnh lên), SFT(10) dịch một bit sang trái. Trạng thái bit đưa vào bit trọng số nhỏ nhất của St là 1 hay 0 tuỳ thuộc vào I On hay Off. Bit trọng số lớn nhất của E sẽ mất đi khi lệnh thực hiện.
Ngõ vào R On lệnh sẽ được reset.

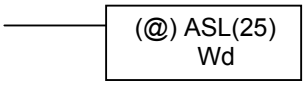
2. Word Shift :



E phải lớn hơn hay bằng St, Evà St phải trên cùng vùng dữ liệu.

WSFT(16) dịch nội dung một word sang trái giữa những word từ St đến E. Nội dung của St sau khi lệnh thực hiện sẽ là 0000, nội dung của E bị mất.

3. Arithmetic Shift Left :

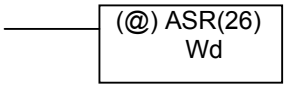


DM 6144 đến DM 6655 không được sử dụng cho Wd.

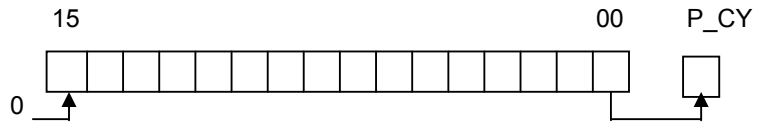
ASL(25) dịch 0 vào bit 00 của Wd, dịch một bit của Wd sang trái, và dịch trạng thái bit 15 vào P_CY.

P_ER : Địa chi gián tiếp của DM không tồn tại.
P_CY : Nhận dữ liệu của bit 00
P_EQ : On khi nội dung của Wd là 0.

4. Arithmetic Shift Right :

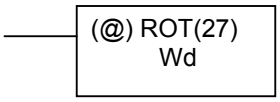


ASR(26) dịch 0 vào bit 15 của Wd trong mỗi chu kỳ quét, dịch một bit của Wd sang phải, và dịch trạng thái bit 00 vào P_CY .

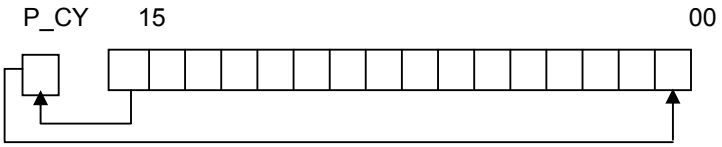


P_ER : On khi địa chỉ gián tiếp của DM không tồn tại.
P_CY : Nhận dữ liệu từ bit 00.
P_EQ : On khi nội dung của Wd là 0.

5. Rotate Left :



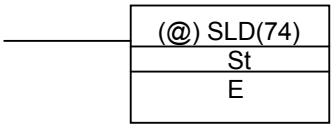
ROL(27) dịch tất cả các bit của Wd sang trái một bit, dịch P_CY vào bit 00, dịch bit15 vào P_CY.



Chú ý :
Dùng STC(41) hay CLC(41) để đặt hoặc xóa P_CY trước khi sử dụng ROT(27) để đảm bảo P_CY chứa trạng thái đúng trước khi lệnh thực hiện.
P_CY sẽ được dịch vào bit 15 trong mỗi chu kỳ quét. Dùng @ hay DIFU/DIFD để chỉ dịch một lần nội dung P_CY vào bit 15.

P_ER : Địa chỉ gián tiếp của DM không tồn tại.
P_CY : Nhận dữ liệu của bit 0.
P_EQ : On kho nội dung của Wd là 0.

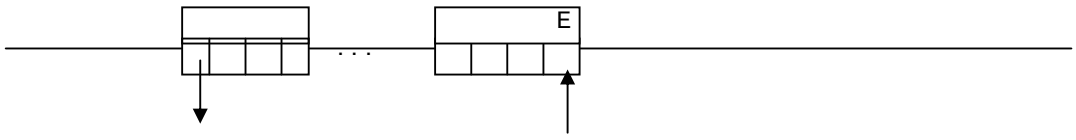
6. One Digit Shift Left :



St : Start Word.
E : End Word.

St và E phải trên cùng vùng dữ liệu, E phải lớn hơn hay bằng St.

SLD(74) dịch dữ liệu giữa St và E một digit (4 bits) sang trái. 0 sẽ được viết vào số có trọng số nhỏ nhất của St, nội dung của số có trọng số lớn nhất của E sẽ bị mất.



Dữ liệu sẽ bị mất 0

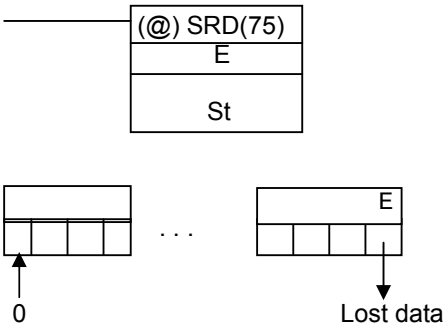
Chú ý :

Khi mất nguồn cung cấp trong lúc SLD(74) thực hiện dịch digit qua 50 word, lệnh có thể thực hiện không hoàn thành.

Bit 0 sẽ được dịch vào bit có trọng số nhỏ nhất của St trong mỗi chu kỳ quét.

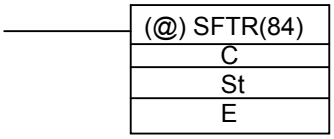
P_ER : khi St và E không trên cùng vùng dữ liệu, địa chỉ gián tiếp của DM không tồn tại.

7. One Digit Shift Right :



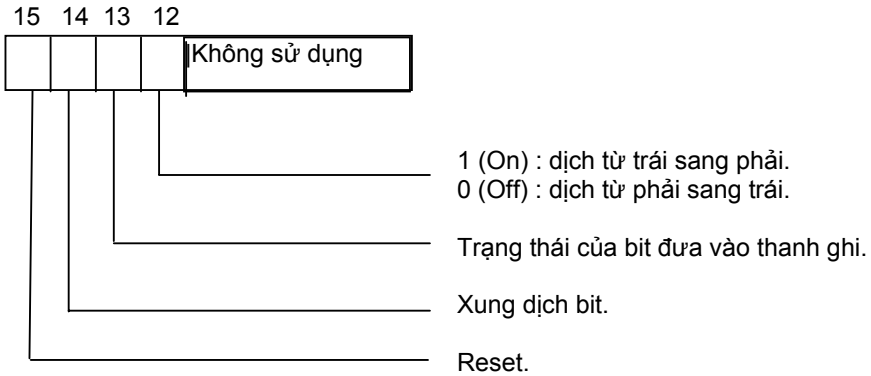
Tương tự SLD(74).

8. Reversible Shift Register :



C : Word điều khiển.
St : Word bắt đầu.
E : Word kết thúc.
St và E phải trên cùng vùng dữ liệu. St phải nhỏ hơn hay bằng E.

Nội dung word điều khiển như sau :

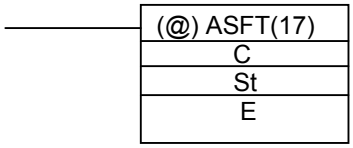


SRD(75) thực hiện dịch bit giữa St và E . Lệnh sẽ dịch trái hay phải, bit đưa vào thanh ghi là 0 hay tuỳ thuộc vào nội dung của word điều khiển C.

P_ER : Khi St và E không cùng nằm trên một vùng dữ liệu.

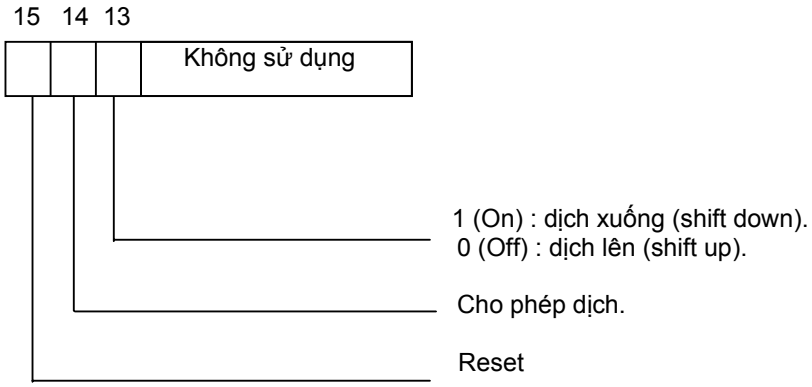
Khi địa chỉ gián tiếp của DM không tồn tại.
P_CY : Nhận trạng thái của bit 00 của St hay bit 15 của bit E tuỳ thuộc vào bit 12 của C.

9. Asynchronous Shift Register :



C : Word điều khiển.
St : Word bắt đầu.
E : Word kết thúc.
St và E phải trên cùng vùng dữ liệu. E phải lớn hơn hay bằng St.

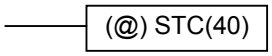
Nội dung của Word điều khiển như sau :



Thanh ghi giữa St và R sẽ có giá trị 0 khi ASFT(17) thực hiện với điều kiện reset On.
ASFT(17) thực hiện dịch ngược không đồng bộ những word trong thanh ghi được xác định bởi St và E. ASFT(17) chỉ thực hiện dịch một word khi word đứng sau nó là 0. Nếu trong thanh ghi không có word nào là 0 thì lệnh sẽ không làm gì cả.
P_ER : Khi St và E không cùng nằm trên một word.
Địa chỉ gián tiếp của DM không tồn tại.

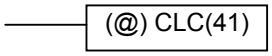
VI- CÁC LỆNH TÍNH đếnÁN DỮ LIỆU DẠNG BCD :

1. Set Carry :



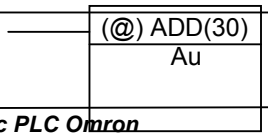
Đặt bit Carry P_CY lên On.

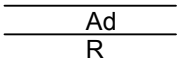
2. Clear carry :



Xoá bit Carry P_CY xuống Off.

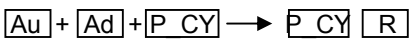
3. BCD Add :





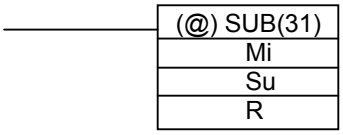
Au : Số được cộng (BCD)
Ad : Số cộng (BCD)
R : Word kết quả.

ADD(30) thực hiện cộng nội dung của Au, Ad, P_CY, kết quả trả về R. P_CY sẽ On khi kết quả lớn hơn 9999.



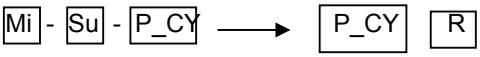
P_ER : Khi nội dung Au và/hoặc Ad không phải dạng BCD.
Khi địa chỉ gián tiếp của DM không tồn tại.
P_CY : On khi có nhớ trong phép cộng.
P_EQ : On khi kết quả là 0.

2. BCD Subtract :



Mi : Số bị trừ. (BCD)
Su : Số trừ. (BCD)
R : Word kết quả.

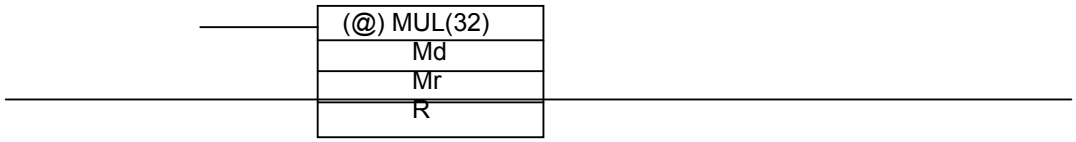
SUB(31) thực hiện trừ nội dung của Mi cho Su và P_CY, kết quả trả về R. Nếu kết quả âm P_CY sẽ On và kết quả trong R chuyển sang dạng bù 10 (10's complement), để có kết quả thật, thực hiện phép trừ 0 cho nội dung của R.



P_ER : Mi và/hoặc Su có nội dung không phải BCD.
Địa chỉ gián tiếp của DM không tồn tại.
P_CY : On khi kết quả âm.
P_EQ : On khi kết quả là 0.

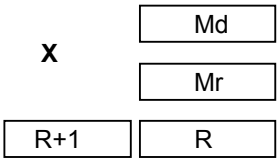
Chú ý : Xoá P_CY trước khi thực hiện SUB(31), kiểm tra P_CY sau khi thực hiện SUB(31) để biết kết quả là âm hay dương .

3. BCD Multiply :



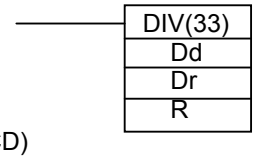
Md : Số được nhân. (BCD)
Mr : Số nhân. (BCD)
R : Wor kết quả.

MUL(32) thực hiện Md ch Mr, kết quả trả về R và R+1.



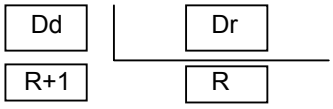
P_ER : Khi nội dung Md và/hoặc Mr không phải dạng BCD.
P_EQ : On khi kết quả là 0.

4. BCD Device :



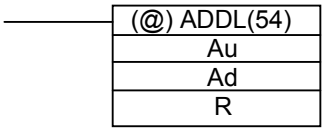
Dd : Số bị chia. (BCD)
Dr : Số chia. (BCD)
R : Word kết quả.

R và R+1 phải trên cùng một vùng dữ liệu.
DIV(33) thực hiện chia Dd cho Dr , kết quả trả về R, số dư trả về R+1.



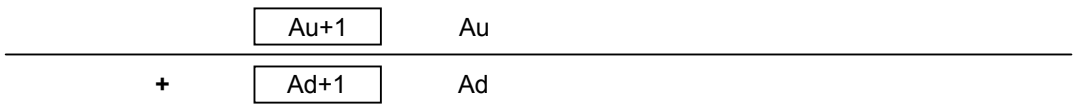
P_ER : Dd hoặc Dr không phải là BCD.
Địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả là 0.

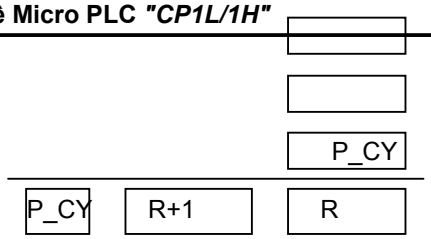
5. Double BCD Add :



Au : Word được cộng thứ nhất.
Ad : Word cộng thứ nhất.
R : Word kết quả thứ nhất.

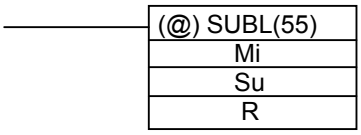
ADDL(54) thực hiện cộng nội dung của P_CY với 8 số được xác định bởi Au, Au+1 và 8 số được xác định bởi Ad, Ad+1 . Kết quả trả về word R, R+1. P_CY On nếu kết quả lớn hơn 99999999.





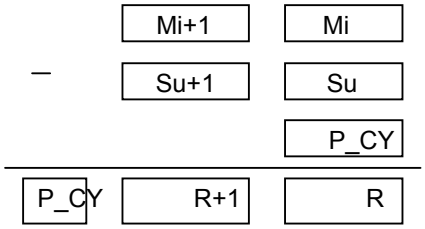
P_ER : Nội dung của Au và/hoặc Ad không phải dạng BCD.
Địa chỉ gián tiếp của DM không tồn tại.
P_CY : On khi kết quả lớn hơn 99999999 (Có nhớ trong phép cộng).
P_EQ : On khi kết quả là 0.

6. Double BCD Subtract :



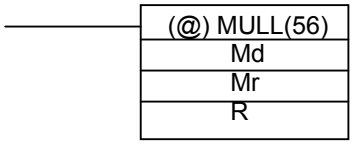
Mi : Word bị trừ đầu tiên. (BCD)
Mu: Word trừ đầu tiên.(BCD)
R : Word kết quả đầu tiên.

SUBL(55) thực hiện phép trừ Mi, Mi+1 cho Mu, Mu+1 và P_CY, kết quả trả về word R, R+1. Nếu kết quả âm, bit P_CY On và kết quả trả về word R, R+1 ở dạng bù 10. Để có được kết quả đúng thực hiện tiếp phép trừ 0 cho R, R+1.
(Chú ý : Sử dụng BSET(71) để tạo ra một hằng 8 digits có giá trị 0)



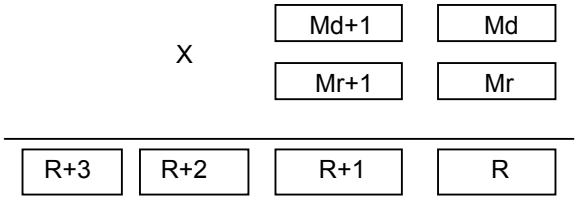
P_ER : Khi nội dung của Mi, Mi+1, Su, Su+1 không phải dạng BCD.
Khi địa chỉ gián tiếp của DM không tồn tại.
P_CY : On khi kết quả âm.
P_EQ : On khi kết quả là 0.

7. Double BCD Multiply :



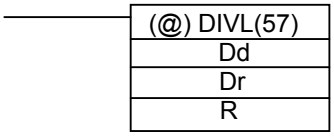
Md : Word được nhân đầu tiên. (BCD)
Mr : Word nhân đầu tiên. (BCD)
R : Word kết quả đầu tiên.

MULL(56) thực hiện nhân nội dung 8 digits của Md, Md+1 với Mr, Mr+1, kết quả trả về word R đến R+3.



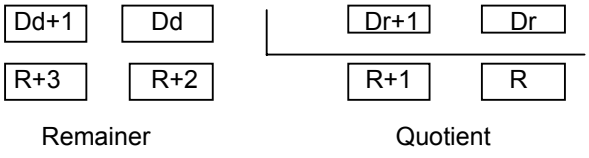
P_ER : Khi nội dung của Md, Md+1, Mr, Mr+1 không phải dạng BCD.
Khi địa chỉ gián tiếp của DM không tồn tại.
P_CY : On khi có nhớ trong kết quả.
P_EQ : On khi kết quả là 0.

8. Double BCD Divide :



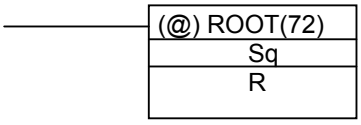
Dd : Word bị chia đầu tiên. (BCD)
Dr : Word chia đầu tiên. (BCD)
R : Word kết quả đầu tiên.

DIVL(57) thực hiện phép chia nội dung 8-digit của Dd, Dd+1 cho nội dung của Dr, Dr+1, kết quả trả về các word từ R, R+1 ; số dư trả về word R+2, R+3.



P_ER : Dr, Dr+1 có nội dung là 0.
Dd, Dd+1, Dr, Dr+1 có nội dung không phải là BCD.
Địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả là 0.

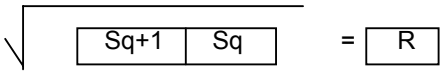
9. Square Root :



Sq : Word nguồn đầu tiên. (BCD)
R : Word kết quả .

Lệnh này không sử dụng cho CPM1, SRM1.

ROOT(72) thực hiện lấy căn bậc 2 nội dung 8-digit của Sq, Sq+1, kết quả trả về R. Phần thập phân của kết quả bị bỏ đi.

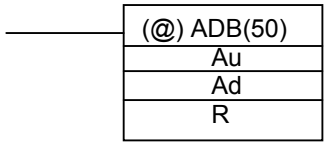


Td : $\sqrt{63250561}$ = 7953.0221...., kết quả được làm tròn 7953.

P_ER : Nội dung của Sq không phải BCD.
Word gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả là 0.

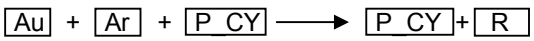
VII- CÁC LỆNH TÍNH đếnÁN DỮ LIỆU DẠNG NHỊ PHÂN (BINARY) :

1. Binary Add :



Au : Word được cộng. (binary)
Ad : Word cộng. (binary)
R : Word kết quả.

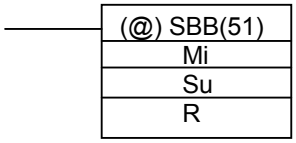
ADB(50) thực hiện phép cộng nội dung dạng nhị phân của Au, Ad, và P_CY, kết quả trả về R. P_CY sẽ ON nếu kết quả lớn hơn FFFF.



ADB(50) còn có thể được sử dụng cộng nội dung nhị phân có dấu. Với CPM1A, SRM1, bit SR 25404 và SR 25405 sẽ tác động khi kết quả vượt ngoài giới hạn trên/dưới (-32 767 đến +32 768) .

P_ER : Địa chỉ gián tiếp của DM không tồn tại.
P_CY : ON khi kết quả lớn hơn FFFF.
P_EQ : ON khi kết quả là 0.
P_OF : ON khi kết quả vượt quá giới hạn +32 767 (7FFF). (Chỉ với CJ1M, CP1L/1H, CJ1, CS1)
P_UF : ON khi kết quả vượt ngoài giới hạn dưới -32 768 (8000). (Chỉ với CJ1M, CP1L/1H, CJ1, CS1)

Binary Subtract :



Mi : Word bị trừ. (Binary)
Su: Word trừ. (Binary)
R : Word kết quả.

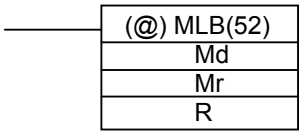
SBB(51) thực hiện trừ nội dung của Mi cho Su và P_CY, kết quả trả về word R. Nếu kết quả âm, P_CY sẽ ON và kết quả trong R có dạng bù 2.

SBB(51) cũng có thể sử dụng thực hiện phép trừ nhị phân có dấu. Với CPM1A, SRM1, bit P_OF & P_UF sẽ tác động nếu kết quả vượt ngoài giới hạn của dữ liệu nhị phân 16-bit có dấu.

P_ER : Địa chỉ gián tiếp của DM không tồn tại.
P_CY : On khi kết quả âm.
P_EQ : On khi kết quả là 0.
P_OF : On khi kết quả vượt quá 32 767 (7FFF). (Chỉ với CJ1M, CP1L/1H, CJ1, CS1)
P_UF : On khi kết quả vượt quá -32 768 (8000). (Chỉ với CJ1M, CP1L/1H, CJ1, CS1)

Chú ý : Để chuyển từ dạng bù 2 sang dạng thông thường, sử dụng lệnh NEG(-).

2. Binary Multiply :

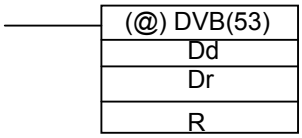


MLB(52) không thể sử dụng nhân dữ liệu nhị phân có dấu. (Lệnh nhân nhị phân có dấu MBS(-) - không sử dụng cho CQM1) .

MLB(52) thực hiện nhân dữ liệu nhị phân của Md với Mr, 4-digit có trọng số thấp đặt vào R, 4-digit có trọng số cao đặt vào R+1.

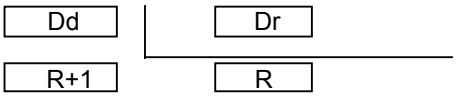
P_ER: Nếu địa chỉ gián tiếp của DM không tồn tại.
P_EQ: On khi kết quả là 0.

3. Binary Divide :



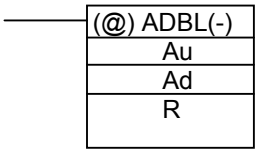
DVB(53) thể thực hiện chia số nhị phân có dấu. (Lệnh chia nhị phân có dấu DBS(-) có thể được sử dụng cho CQM1).

DVB(53) thực hiện chia nội dung của Dd cho Dr, kết quả trả về R , số dư trả về R+1.



P_ER: Nội dung của Dr là 0.
Địa chỉ gián tiếp của DM không tồn tại.
P_EQ: On khi kết quả là 0.

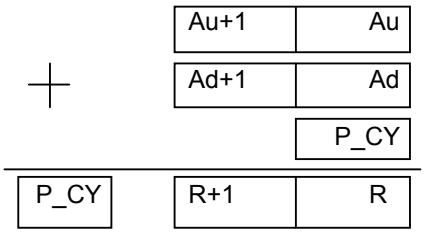
4. Double Binary ADD :



Au: Word được cộng thứ nhất. (Binary)
Ad: Word cộng thứ nhất. (Binary)
R : Word kết quả thứ nhất.

Lệnh này chỉ sữ dụng cho CQM1-CPU4 - EV1.

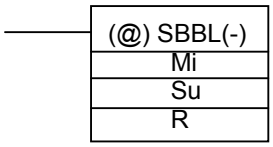
Au, Au+1 ; Ad, Ad+1 phải trên cùng vùng dữ liệu,
ADBL(-) thực hiện cộng nội dung 8-digit của Au+1, Au với nội dung 8-digit của Ad+1, Ad, và P_CY, kết quả trả về R+1, R. P_CY sẽ tác động nếu kết quả lớn hơn FFFF FFFF.



ADBL(-) có thể được sử dụng cộng dữ liệu nhị phân có dấu. Bit SR 25404 hoặc SR 25405 sẽ tác động nếu kết quả vượt ngoài giới hạn của dữ liệu 32-bit.

P_ER: Địa chỉ gián tiếp DM không tồn tại.
P_CY: On khi kết quả lớn hơn FFFF FFFF
P_EQ: On khi kết quả là 0.
P_OF: On khi kết quả lớn hơn + 2 147 483 647 (7FFF FFFF).
P_UF: On khi kết quả nhỏ hơn - 2 147 483 648 (8000 000).

5. Double Binary Subtract :



Mi: Word bị trừ. (Binary)
Su: Word trừ. (Binary)
R: Word kết quả.

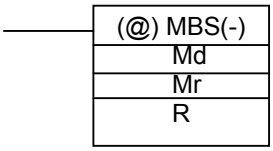
Lệnh này chỉ sử dụng cho PLC loại CJ1M, CP1L/1H, CJ1 & CS1.
Mi và Mi+1, Su và Su+1, R và R+1 phải trên cùng vùng dữ liệu.

SBBL(-) thực hiện trừ nội dung của Mi+1, Mi cho Su+1, Su và P_CY, kết quả trả về R+u kết quả âm, P_CY sẽ On, nội dung trong R+1, R có dạng bù 2. Sử dụng NEG(-) chuyển từ dạng bù 2 sang kết quả thực.
SBBL(-) cũng có thể sử dụng cho phép trừ dạng nhị phân có dấu. Bit SR 25404 hoặc SR 25405 sẽ tác động nếu kết quả vượt ngoài giới hạn của dữ liệu nhị phân 32-bit.

P_ER: Địa chỉ gián tiếp của DM không tồn tại.
P_CY: On khi kết quả âm.
P_EQ: On khi kết quả là 0.
P_OF: On khi kết quả lớn hơn + 2 147 483 647 (7FFF FFFF).
P_UF: On khi kết quả nhỏ hơn - 2 147 483 648 (8000 0000).

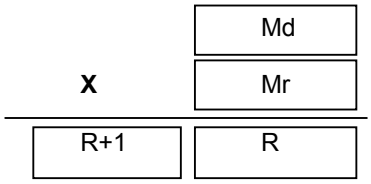
Chú ý: 1. Đối với dữ liệu nhị phân không dấu, P_CY dùng để chỉ ra kết quả âm. Dùng lệnh NEGL(-) để chuyển kết quả dạng bù 2 sang dạng thực.
2. Đối với dữ liệu nhị phân có dấu, bit P_OF , P_UF để chỉ kết quả vượt ngoài khoảng (- 2 147 483 647 , + 2 147 483 647).

6. Signed Binary Multiply :



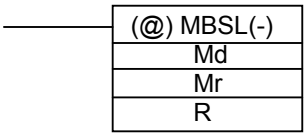
Lệnh này chỉ sử dụng cho PLC loại CJ1M, CP1L/1H, CJ1 & CS1.

MBS(-) nhân nội dung nhị phân có dấu của hai word Md, Mr, kết quả là 8-digit binary có dấu trả về word R+1, R.



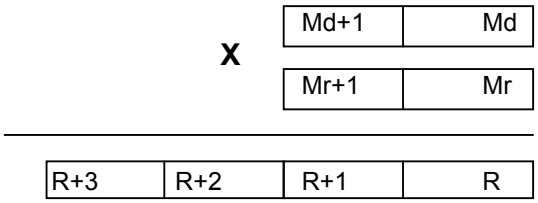
P_ER: Địa chỉ gián tiếp của DM không tồn tại.
P_EQ: On khi kết quả là 0.

7. Double Signed Binary Multiply :



Lệnh này chỉ sử dụng cho PLC loại CJ1M, CP1L/1H, CJ1 & CS1.
Md và Md+1, Mr và Mr+1 phải trên cùng vùng dữ liệu. R đến R+3 phải trên cùng vùng dữ liệu.

MBSL(-) thực hiện nhân nội dung 32-bit (8-digit) dạng nhị phân có dấu của Md+1,Md với nội dung 32-bit dạng nhị phân có dấu của Mr+1,Mr . Kết quả dạng nhị phân có dấu 16-bit trả về word R+3 đến R.



P_ER : Khi địa chỉ gián tiếp của DM không tồn tại.
P_EQ : ON khi kết quả là 0.

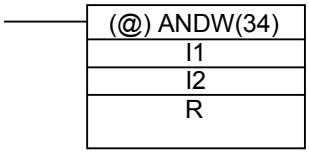
VIII- LỆNH LOGIC :

1. Complement :

COM(29) xoá tất cả bit On về OFF và đặt tất cả bit OFF lên ON.

P_ER : Khi địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả là 0.

2. Logical AND :

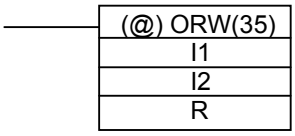


I1 : Input 1
I2 : Input 2
R : Word kết quả.

ANDW(34) thực hiện logic AND từng bit trong nội dung hai word I1 và I2 , kết quả trả về R.

P_ER : Khi địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả bằng 0.

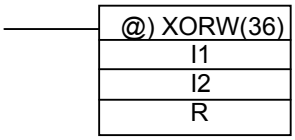
3. Logical OR :



I1 : Input 1
I2 : Input 2
R : Word kết quả.
ORW(35) thực hiện logic OR từng bit giữa nội dung của hai word I1 và I2, kết quả trả về R.

P_ER : Khi địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả bằng 0.

4. Exclusive OR :

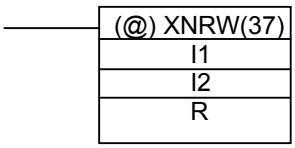


I1 : Input 1
I2 : Input 2
R : Word kết quả.

ORW(35) thực hiện logic Exclusive OR từng bit giữa nội dung của hai word I1 và I2, kết quả trả về R.

P_ER : Khi địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả bằng 0.

5. Exclusive NOR :

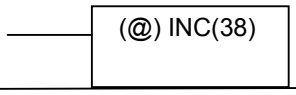


I1 : Input 1
I2 : Input 2
R : word kết quả.
ORW(35) thực hiện logic Exclusive NOR từng bit giữa nội dung của hai word I1 và I2, kết quả trả về R.

P_ER : Khi địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả bằng 0.

IX- LỆNH TĂNG / GIẢM :

1. BCD Increment :

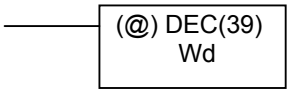


Wd

INC(38) thực hiện tăng nội dung BCD của Wd, kết quả không bị ảnh hưởng bởi P_CY.

P_ER : Nội dung trong WD không phải dạng BCD.
Khi địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả bằng 0.

2. BCD Decrement :



INC(38) thực hiện giảm nội dung BCD của Wd, kết quả không bị ảnh hưởng bởi P_CY.

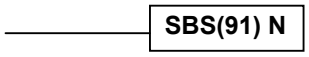
P_ER : Nội dung trong WD không phải dạng BCD.
Khi địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả bằng 0.

X - LỆNH CHƯƠNG TRÌNH CON (Subroutine Instructions):

Chương trình con thực hiện rẽ nhánh chương trình chính trong trường hợp cần thực hiện một hay một nhóm điều khiển vào bất kỳ thời điểm nào trong chu kỳ quét của chương trình chính. Chương trình con có thể được thực hiện một hay nhiều lần trong một chu quét của chương trình chính. Việc viết các lệnh trong chương trình con giống như đối với chương trình chính.

Khi tất cả các lệnh của chương trình con thực hiện xong, chương trình sẽ quay về vị trí ngay sau vị trí gọi chương trình con, chương trình sẽ tiếp tục thực hiện những bước kết tiếp.

1. Subroutine enter - SBS(91):

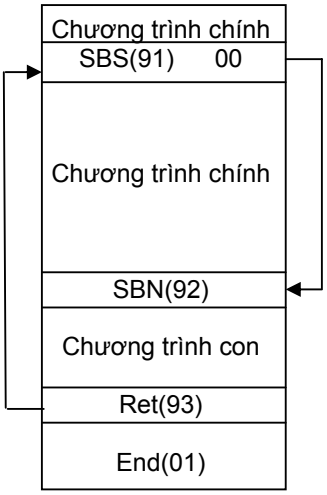


N : số thứ tự của chương trình con. (000 ~ 255).

CJ1M : N= 000 ~ 127.
CPM1 : N= 000 ~ 049.

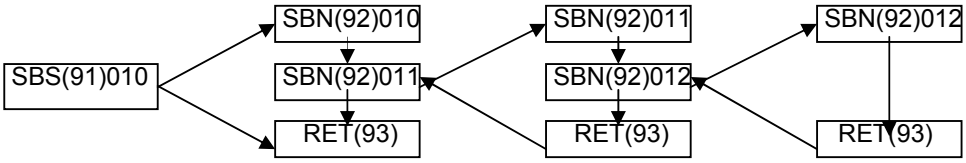
Việc rẽ nhánh chương trình có thể được thực hiện bằng cách đặt SBS(91) vào chương trình chính tại nơi cần rẽ nhánh. Số thứ tự N chỉ ra chương trình con tương ứng sẽ được gọi.

Khi điều kiện cho SBS(91) On, những lệnh giữa SBN(92) có cùng số thứ tự N và RET(93) đầu tiên sẽ được thực hiện trước, sau đó chương trình sẽ quay về thực hiện tiếp các lệnh ngay sau ngay sau SBS(91) vừa gọi.



SBS(91) có thể được dùng nhiều lần trong chương trình, có nghĩa là cùng một chương trình con có thể được gọi nhiều lần tại nhiều nơi trong chương trình chính.

SBS(91) có thể đặt trong một chương trình con để gọi tiếp một chương trình con khác, có nghĩa là chương trình con có thể đặt lồng vào nhau. Việc lồng chương trình con cho phép đến 16 mức. Khi một chương trình con kết thúc, nó sẽ quay về chương trình con có mức cao hơn đã gọi nó.



Bit P_ER ON khi :

Số thứ tự thứ tự của chương trình con không tồn tại.
Chương trình con tự gọi nó.
Gọi một chương trình con đang thực hiện.

Chú ý: SBS(91) sẽ không thực hiện và chương trình con sẽ không được gọi khi P_ER ON.

2. Subroutine define and Return - SBN(92) / RET(93):

SBN(92) N — N : Số thứ tự chương trình con. (000 ~ 255)

RET(93)

Điều kiện :

CJ1M: N=000 ~ 127
CPM1: N=000 ~ 049
CP1L/1H: N=0 ~ 255

N chỉ được sử dụng trong SBN(92) một lần.

SBN(92) là điểm bắt đầu một chương trình con, RET(93) là điểm kết thúc. Mỗi chương trình con được nhận dạng bởi N.

Tất cả chương trình con phải được lập trình ở cuối chương trình chính.Chương trình chính sẽ thực hiện một hay nhiều chương trình con (nếu nó được gọi) trước khi trở về địa chí 0000 để thực hiện chu kỳ quét kế tiếp.

END(01) phải được đặt ngay sau RET(93) cuối cùng.

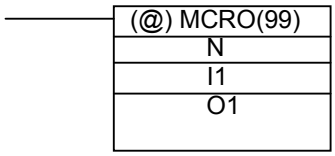
Chú ý:

Trong một chu kỳ quét, chương trình sẽ quay trở về địa chỉ đầu tiên để thực hiện chu kỳ quét kế tiếp nếu gặp SBN(92).

Nếu DIFU(13) hay DIFD(14) được đặt trong chương trình con, bit tác động bởi hai lệnh trên chỉ OFF khi chương trình con được gọi lại lần thứ hai, có nghĩa là thời gian ON kéo dài hơn một chu kỳ.

XI - LỆNH ĐẶC BIỆT (Special Instructions):

1. Macro - MCRO(99)



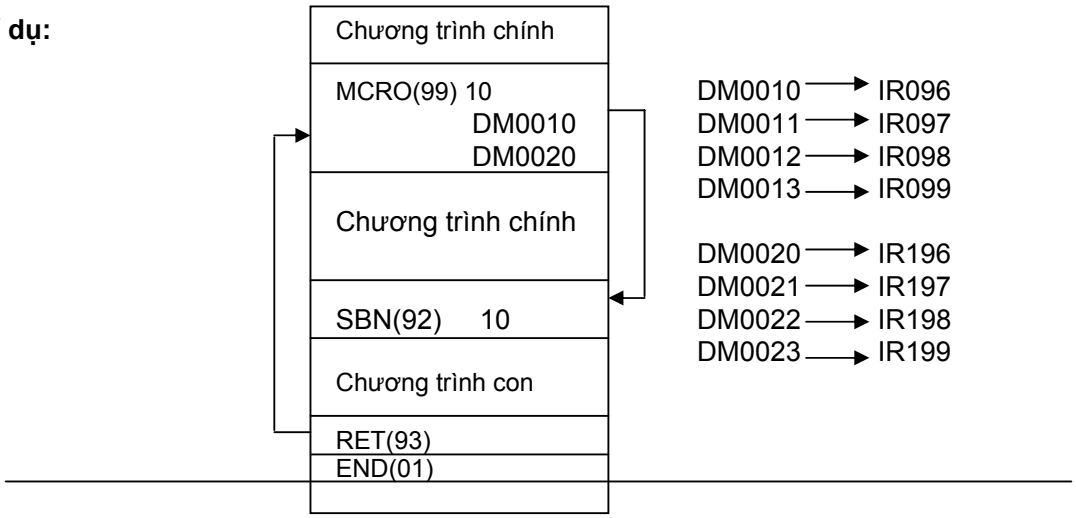
N: Số thứ tự chương trình con (000 ~ 127)
I1: Word input đầu tiên.
O1: Word output đầu tiên.

Điều kiện:

MCRO(99) cho phép một chương trình con có thể thực hiện nhiều chức năng khác nhau. Có nghĩa là một chương trình con có thể thay thế cho nhiều chương trình con khác có cấu trúc giống nhau nhưng kết quả hoạt động khác nhau. Có 04 word input, IR096 ~ IR099 (IR232 ~ IR235 đối với CPM1), và 04 word output, IR196 ~ IR199 (IR236 ~ IR239 đối với CPM1) được dùng cho MCRO(99). 08 word này được dùng trong chương trình con và dữ liệu của nó là được lấy từ các word I1 ~ I1+3 và O1 ~ O1+3 khi chương trình con làm việc.

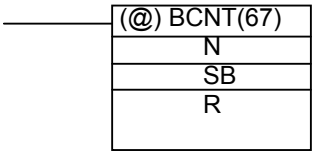
MCRO(99) copy nội dung của I1 ~ I1+3 vào IR096 ~ IR099, nội dung của O1 ~ O1+3 vào IR196 ~ IR199, sau đó gọi và thực hiện chương trình con N. Khi thực hiện xong chương trình con, nội dung của IR196 ~ IR199 được truyền trở lại O1 ~ O1+3.

Ví dụ:



P_ER On khi: Chương trình con hay N không tồn tại .
Vùng dữ liệu nằm ngoài vùng cho phép.
Địa chỉ tương đối của DM không tồn tại.
Chương trình con tự gọi nó.
Gọi một chương trình con đang làm việc.

2. Bit Counter - BCNT(67)



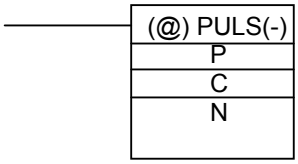
N: Số lượng word (BCD).
SB: Word nguồn đầu tiên.
R: Wor kết quả đầu tiên.

Điều kiện: N phải khác 0.

BCNT(67) đếm tất cả số lượng bit ở trạng thái ON trong tất cả các word từ SB đến SB+(N-1), kết quả được trả về R.
P_ER : N không phải là BCD hay N=0: SB ~ SB+(N-1) không ở trên cùng vùng dữ liệu.
Kết quả trong R lớn hơn 9999.
Địa chỉ gián tiếp DM không tồn tại.
P_EQ: Khi kết quả là 0. (R : 0000)

XII- CHỨC NĂNG NGÕ RA PHÁT XUNG

1- SET PULSES - PULS(-)



P: Port phát xung (000, 001, 002)
C: Dữ liệu điều khiển. (000 đến 005)
N: Số lượng xung (IR, SR, AR, DM, HR, LR).

Điều kiện: N và N+1 phải trên cùng vùng dữ liệu.

PULS(-) dùng để đặt thông số cho ngõ ra phát xung mà nó được thực hiện bởi SPED(-) hoặc ACC(-).

- **Chọn ngõ ra phát xung:**

P = 000 : Ngõ ra xung là bit output.
P = 001 : Ngõ ra xung là Port 1
P= 002 : Ngõ ra xung là Port 2.

- **Dữ liệu điều khiển C:**

C	Chiều phát xung	Số xung được phát	Thời điểm bắt đầu cạnh xuống
000	Chiều thuận	Đặt trong N và N+1	Không dùng
001	Chiều nghịch	Đặt trong N và N+1	Không dùng
002	Chiều thuận	Đặt trong N và N+1	Đặt trong N+2 và N+3
003	Chiều nghịch	Đặt trong N và N+1	Đặt trong N+2 và N+3
004	Chiều thuận	Không dùng	Không dùng
005	Chiều nghịch	Không dùng	Không dùng

Việc đặt chiều phát xung có tác dụng cho đến khi dừng chương trình hoặc PULS(-) được thực hiện trở lại.

- **Số xung và thời điểm bắt đầu cạnh xuống :**

Khi C = 000 hay 003, N+1 và N chứa giá trị số xung phát ra không phụ thuộc vào chế độ phát xung. N+1, N chứa giá trị từ 00000000 ~ 16777215. Xung sẽ được phát ra khi điều kiện lệnh SPED(-) hay ACC(-) cho phép và sẽ tự động dừng khi số lượng xung phát ra đạt đến số lượng đã đặt trước.

	Leftmost 4 digits	Rightmost 4 digits	Possible range
Số lượng xung được phát:	N	N	00000001 ~ 16777215

Khi C = 002 hay 003, N+3 và N+2 chứa giá trị đặt số lượng xung để thực hiện cạnh xuống trong lệnh ACC(-) ở Mode 0. N+3, N+2 có thể chứa giá trị từ 00000001 ~ 16777215. Xung ra bắt đầu bởi ACC(-) sẽ bắt đầu thực hiện cạnh xuống khi số xung đạt đến giá trị đặt ban đầu.

	Leftmost 4 digits	Rightmost 4 digits	Possible range
Thời điểm thực hiện cạnh xuống	N+3	N+4	00000001 ~ 16777215

Phụ lục 2

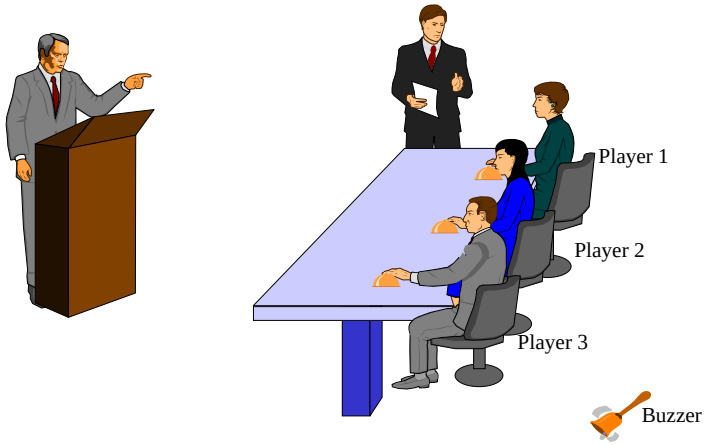
Các ví dụ ứng dụng khác trên bộ CP1L/1H Training Kit

Chúng ta sẽ cùng xem xét thêm một số ứng dụng mô phỏng trên bộ CP1L/1H Training Kit với chương trình được lập và in ra bằng CX-Programmer.

1. Chương trình điều khiển trò chơi dạng "Đường lên đỉnh Olympia"

<Mô tả yêu cầu >

Sau khi người dẫn chương trình (Host) đã nêu xong các câu hỏi, các đấu thủ (player) sẽ bấm nút phía trước mặt để trả lời câu hỏi. Ai bấm trước trả lời trước. Chuông (Buzzer) sẽ kêu trong 10 giây sau khi bất kỳ đấu thủ nào bấm nút. Cùng lúc đó đèn trước mặt đấu thủ đó sẽ sáng và sẽ chỉ được tắt (Reset) bởi người dẫn chương trình.



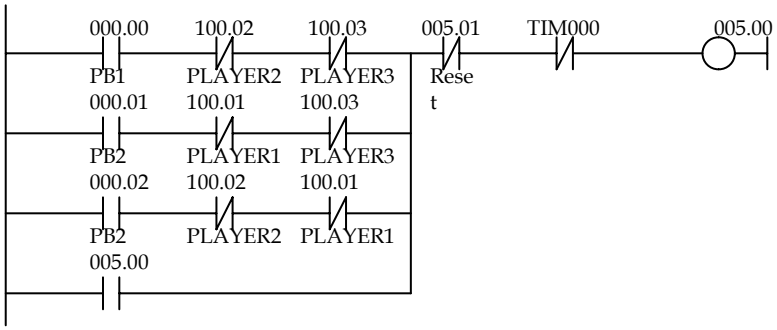
Các đầu vào ra

Đầu vào			***Đầu ra***		
00000	-	Nút bấm đấu thủ 1 (PB1)	100.00	-	Còi
00001	-	Nút bấm đấu thủ 2 (PB2)	100.01	-	Đèn của Đấu thủ 1
00002	-	Nút bấm đấu thủ 3 (PB3)	100.02	-	Đèn của Đấu thủ 2
00003	-	Nút tắt (Reset)	100.03	-	Đèn của Đấu thủ 3

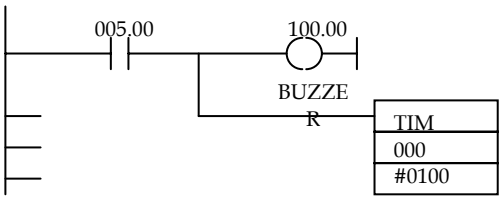
Ladder Diagram : Main 1 Rung 1

Main 1 - Who press first
(Priority Determination)
This program is to determine which player press the switch first, after the host have finished asking a question.

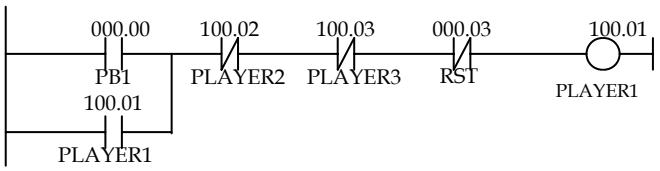
Rung 1 - Interlocked
Interlocked Rung for 3 player playing the game



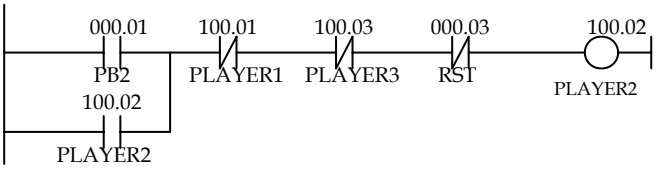
Rung 2 - Buzzer
ON Buzzer when any switch is pressed and timer will cut the buzzer after specified time



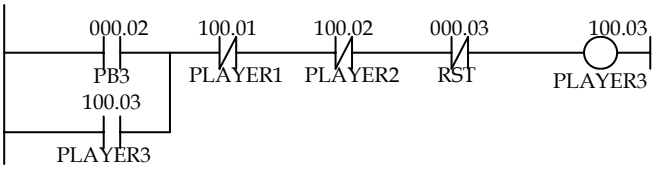
Rung 3 - Player 1
Player 1 Rung



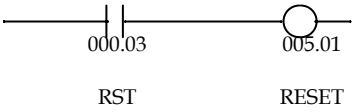
Rung 4 - Player 2
Player 2 Rung



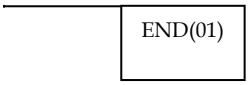
Rung 5 - Player 3
Player 3 Rung



Rung 6 - Reset
Reset for the Game



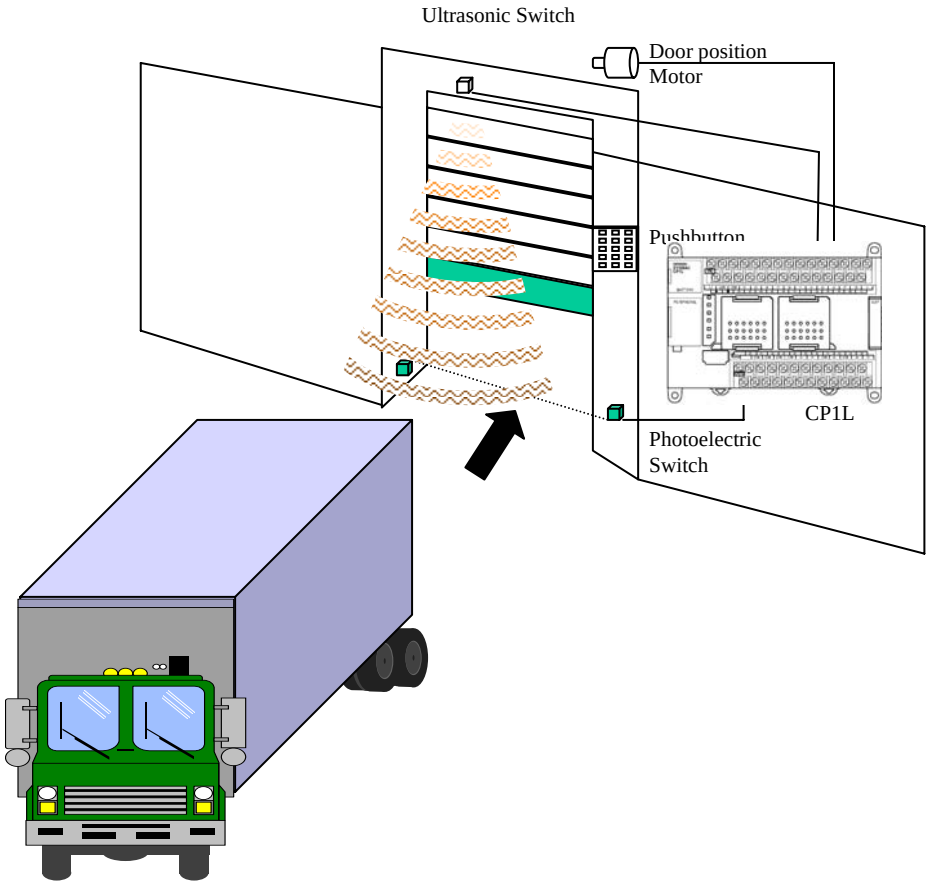
Rung 7



2. Điều khiển đóng mở cửa gara ôtô

<Yêu cầu>

Một cảm biến siêu âm (ultrasonic switch) được dùng để phát hiện ô tô đang lại gần cửa.
Một cảm biến quang điện được dùng để phát hiện ô tô đang đi qua cửa.
PLC sẽ nhận các tín hiệu vào này và điều khiển động cơ đóng mở cửa.



Phụ lục 2

Các đầu vào ra

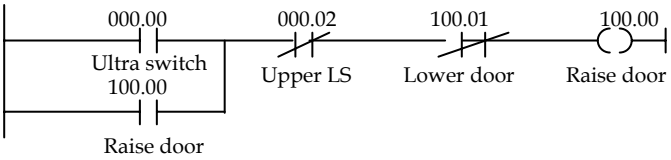
Input	Thiết bị ngoài
00000	Ultrasonic switch
00001	Photoelectric switch
00002	Door Upper limit switch
00003	Door Lower limit switch

Output	Thiết bị ngoài
100.00	Motor to raise door
100.01	Motor to lower door

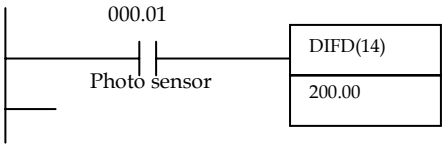
Ladder Diagram : Main 1 Rung 1

Main 1 - Auto door
This program shows the automatic control of warehouse door.

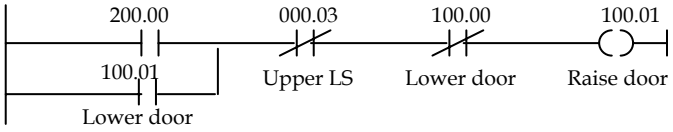
Rung 1 - Raise door



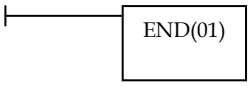
Rung 2 - Photo sensor
Sense unit differentiation down



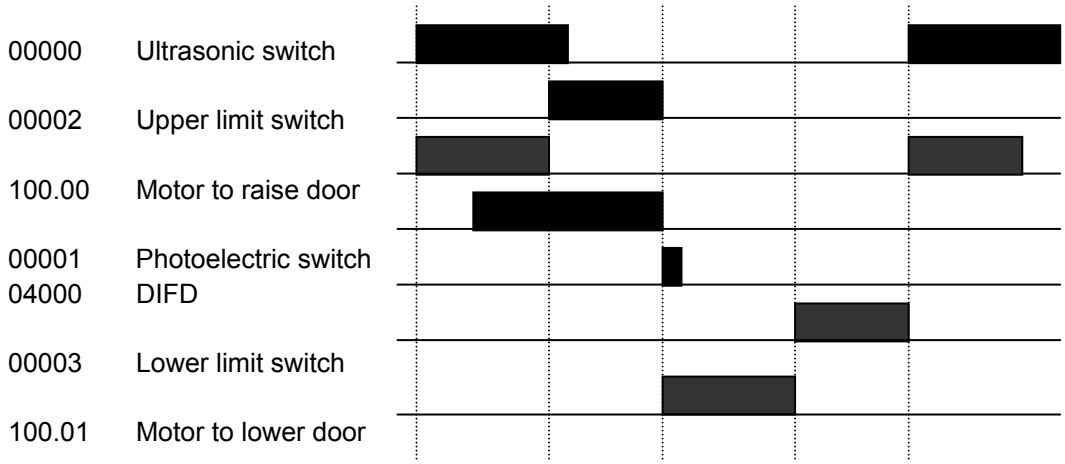
Rung 3 - Lower door



Rung 4 - End



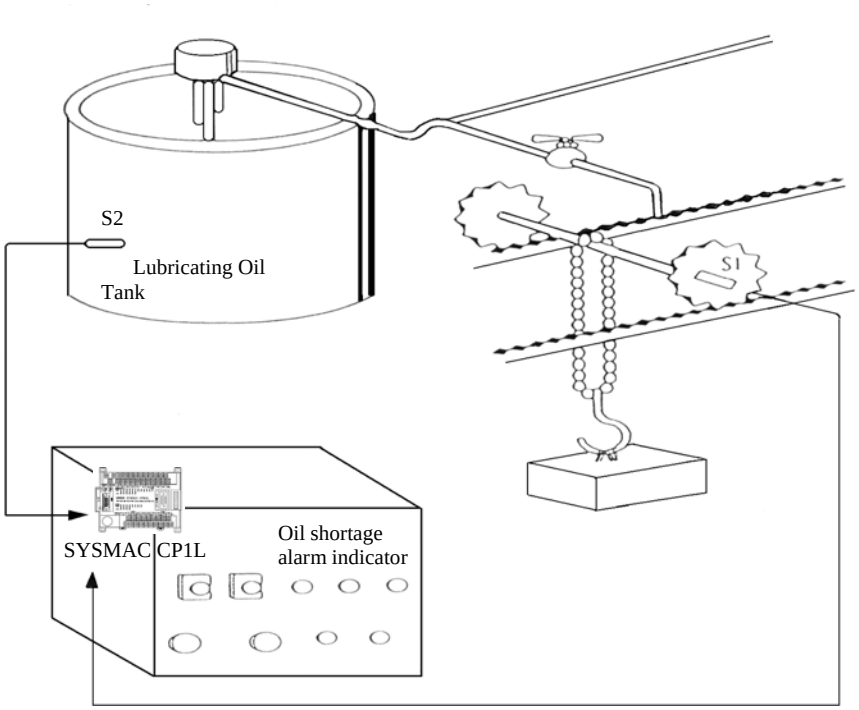
Timing diagram



3. Tự động bôi trơn (Lubrication) dầu cho bánh xe

<Mô tả>

Khi bánh xe di chuyển về phía cảm biến S1, S1 sẽ phát hiện bánh xe và sẽ ra tín hiệu cho van điện từ (Valve V1) để cấp dầu bôi trơn cho bánh xe. Van V1 sẽ mở trong một khoảng thời gian ngắn để cấp một lượng dầu định trước cho bánh xe. Khi cảm biến S2 phát hiện mức dầu trong bồn chứa (Tank) thấp, nó sẽ ra tín hiệu cảnh báo.



Các đầu vào ra

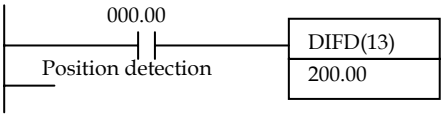
Input	*Thiết bị ngoài*
00000	Position detection (S1)
00001	Lower limit of level (S2)

Output	*Thiết bị ngoài*
100.00	Electromagnetic valve for oil supply
100.01	Oil shortage alarm indicator

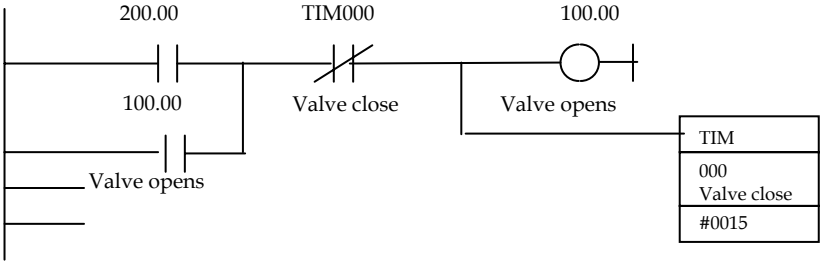
Ladder Diagram : Main 1 Rung 1

Main 1 - Auto lubricate
Auto lubrication of gear

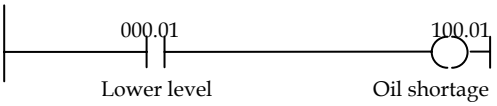
Rung 1 - Start



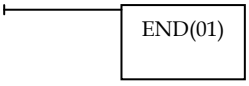
Rung 2 - Open valve
Open valve and delay 1.5 sec.



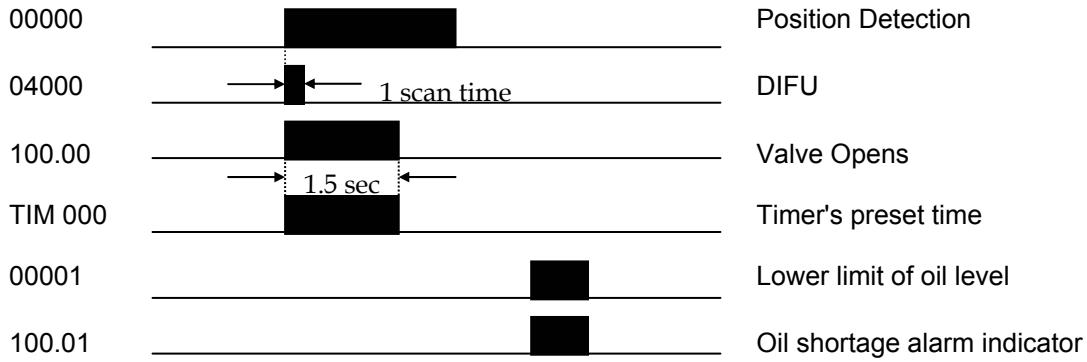
Rung 3 - Oil shortage



Rung 4 - End



Timing diagram



4. Điều khiển động cơ băng tải

Băng tải gồm có 3 phân đoạn, và cần điều khiển sao cho động cơ của mỗi phân đoạn chỉ chạy khi có đối tượng (tấm đồng- copper plate) đang nằm trên phân đoạn tương ứng. Vị trí của tấm kim loại được xác định bởi các cảm biến tiệm cận đặt gần nó (Sensor 1,2,3). Khi tấm kim loại nằm trong khoảng cách phát hiện của 1 sensor, động cơ tương ứng sẽ vẫn làm việc. Khi tấm kim loại nằm ngoài khoảng cách phát hiện của sensor, một timer trễ sẽ được kích hoạt và khi thời gian đặt của timer hết, động cơ tương ứng sẽ ngừng.

I/O

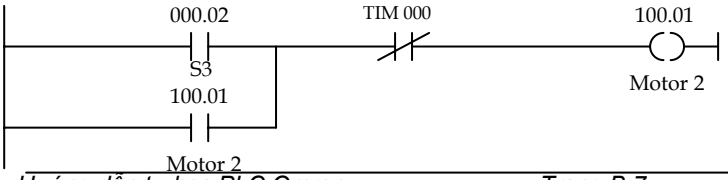
Input	*Thiết bị ngoài*
00000	Sensor 1
00001	Sensor 2
00002	Sensor 3

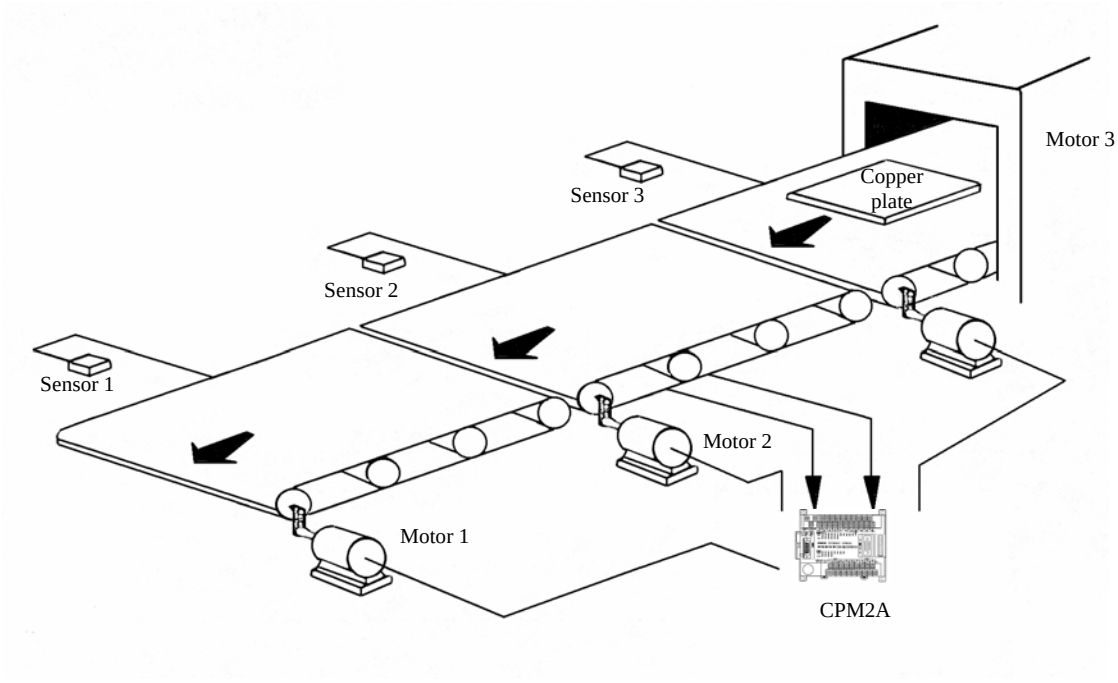
Output	*Thiết bị ngoài*
100.00	Motor 1
100.01	Motor 2
100.02	Motor 3

Ladder Diagram : Main 1 Rung 1

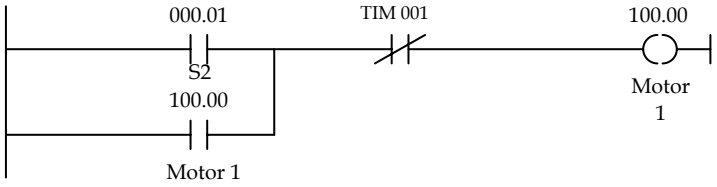
Main 1 - Conveyor control
Conveyor belt control application

Rung 1 - Motor 2

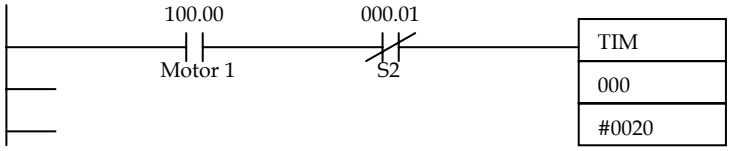




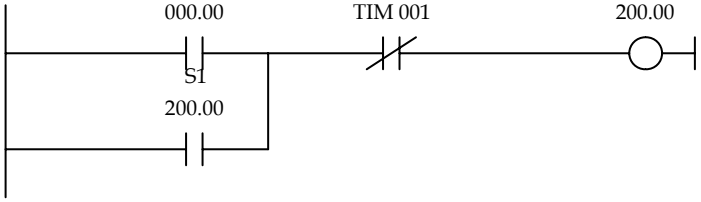
Rung 2 - Motor 1



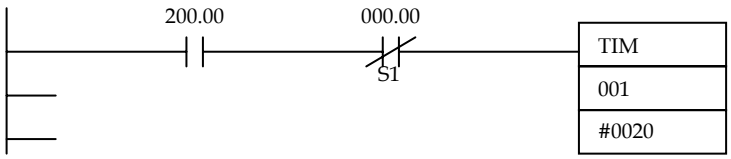
Rung 3 - Delay for 2 sec



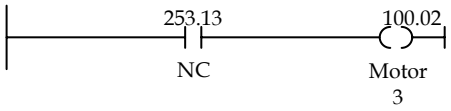
Rung 4 - Sensor 1



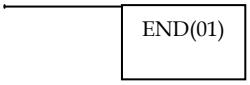
Rung 5 - Delay for 2 sec



Rung 6 - Motor 3



Rung 7 - End



5. Điều khiển dây chuyền đóng gói (Packing Line Control)

Yêu cầu :

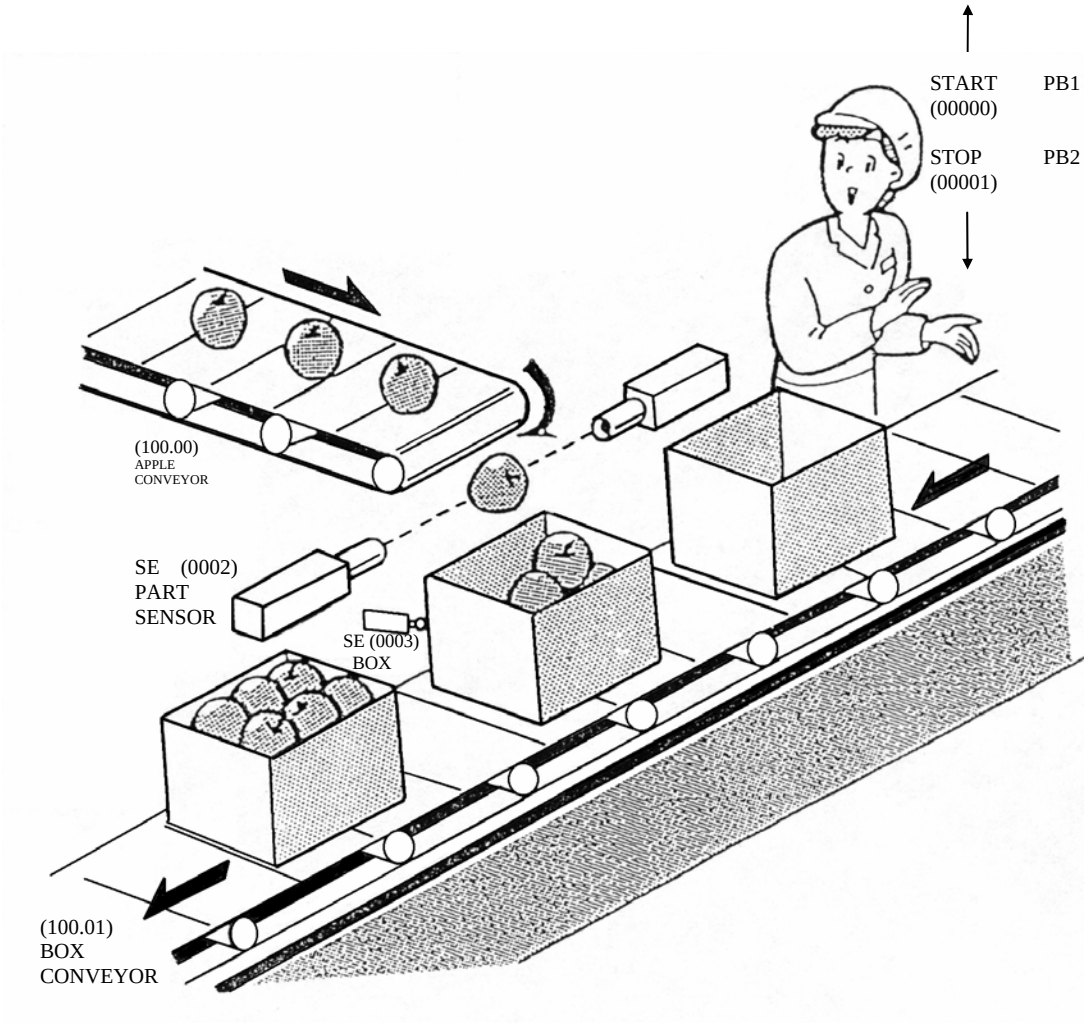
Khi nút bấm PB 1 (Start) được bấm, băng tải hộp bắt đầu chuyển động. Khi phát hiện sự có mặt của hộp, băng tải hộp (Box Conveyor) dừng và băng tải táo (Apple conveyor) bắt đầu chuyển động.

Cảm biến đếm (SE1) sẽ đếm số lượng quả táo cho đến khi đạt 10 quả. Băng tải táo lúc này sẽ dừng và băng tải hộp lại khởi động lại. Bộ đếm sẽ được reset và hoạt động lại lặp lại cho đến khi nút PB2 (Stop) được bấm.

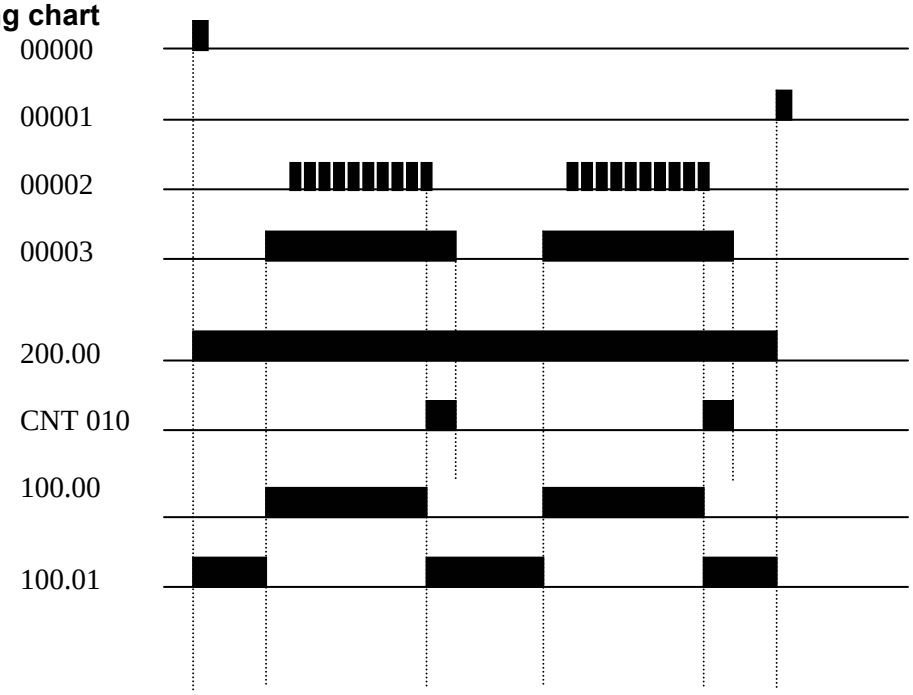
Các đầu vào ra

Input	*Thiết bị ngoài*
00000	START Push button (PB1)
00001	STOP Push button (PB2)
00002	Part Present (SE1)
00003	Box Present (SE2)

Output	*Thiết bị ngoài*
100.00	Apple Conveyor
100.01	Box Conveyor



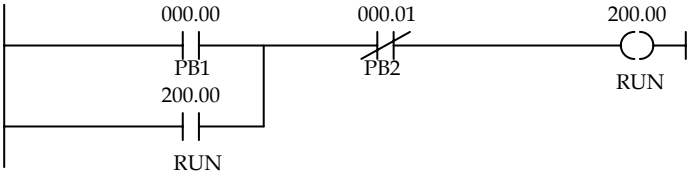
Timing chart



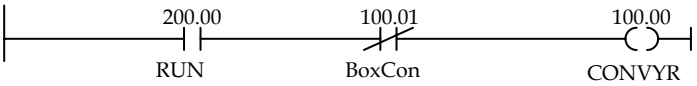
Ladder Diagram : Main 1 Rung 1

Main 1 - Packing
Packing line control for Apples

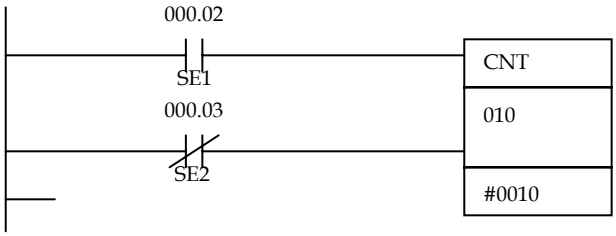
Rung 1 - Start condition



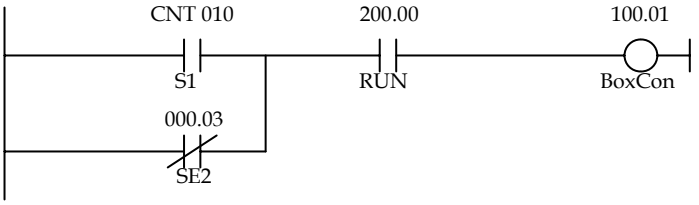
Rung 2 - Apples conveyor



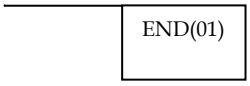
Rung 3 - Counter
Counter preset at 10



Rung 4 - Box conveyor



Rung 5 - END



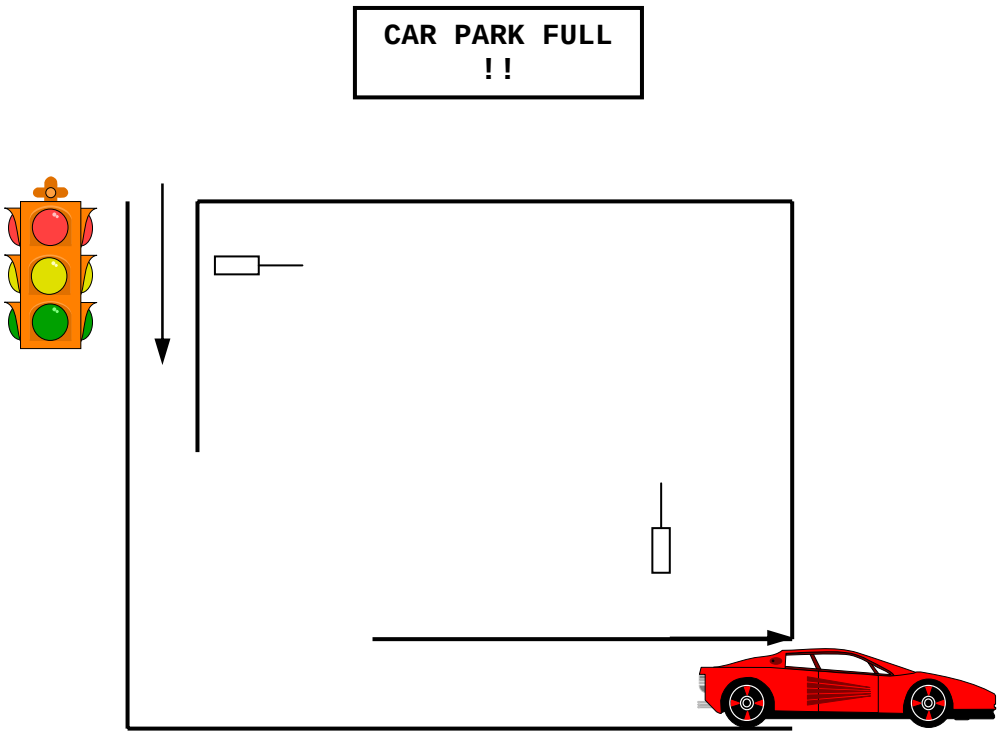
Mnemonic Codes

Địa chỉ	*Lệnh*	*Th. số*
0000	LD	00000
0001	OR	20000
0002	AND NOT	00001
0003	OUT	200.00
0004	LD	200.00
0005	AND NOT	100.01
0006	OUT	100.00
0007	LD	00002

Địa chỉ	*Lệnh*	*Th. số*
0008	LD NOT	00003
0009	CNT	010
		# 0010
0010	LD CNT	010
0011	OR NOT	00003
0012	AND	200.00
0013	OUT	100.01
0014	END (01)	

1.6 Điều khiển bãi đỗ xe

Đây là một chương trình điều khiển bãi đỗ xe đơn giản chỉ cho phép tối đa là 100 xe được đỗ tại một thời điểm. Mỗi khi có một xe mới đi vào, Sensor (S1) sẽ phát hiện và PLC sẽ cộng 1 vào tổng số xe hiện đang trong bãi đỗ và sẽ trừ đi 1 khi Sensor (S2) phát hiện có xe đi ra khỏi bãi đỗ. Khi 100 xe đã đỗ trong bãi, đèn hiệu "CAR PARK FULL" sẽ sáng để báo các xe khác không được vào bãi.



Car coming in

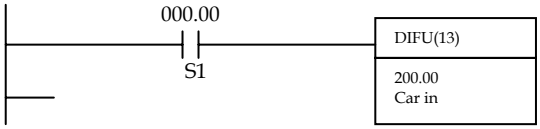
I/O

INPUT	
00000	Sensor S1
00001	Sensor S2

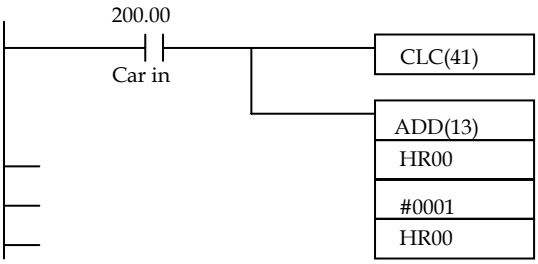
OUTPUT	
100.00	Car park full sign

Ladder Diagram : Main 1 Rung 1
Main 1 - Car Park Control
Application: Car Park Control

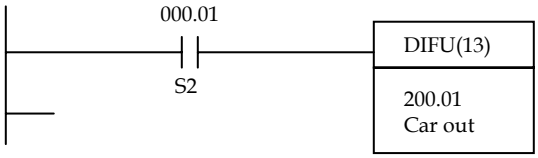
Rung 1 - Car in



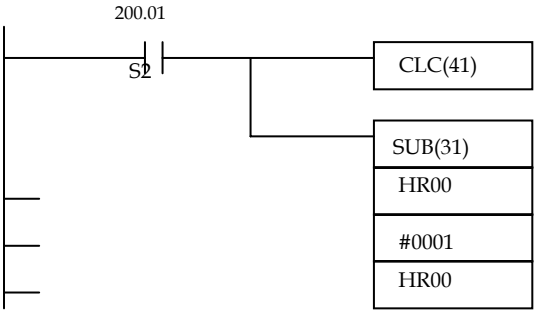
Rung 2 - Add 1



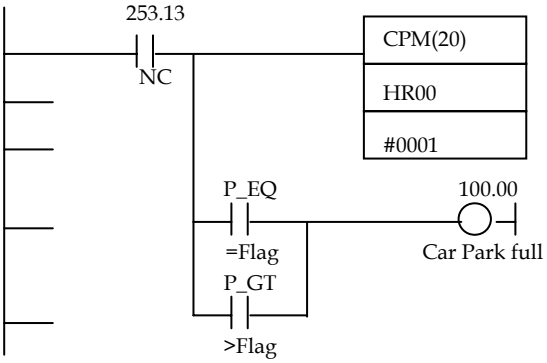
Rung 3 - Car out



Rung 4 - Subtract 1



Rung 5 - Compare



Rung 6 - End