CHƯƠNG 1

GIỚI THIỆU VỀ MICRO PLC "CPM2A"

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 10 chữ số từ 0-9, 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 :

Bảng 1

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

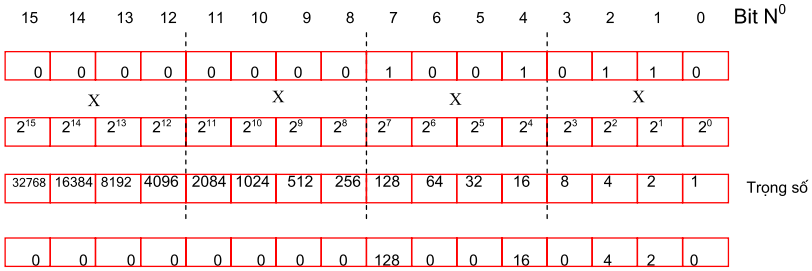
Bảng trên là cách biểu diễn của các chữ số hexa và BCD bằng các chữ số nhị phân (mỗi chữ số hexa và BCD đều cần 4 bit nhị phân).

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)

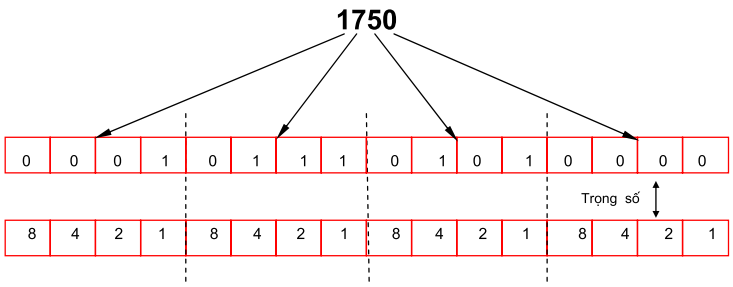
Ngược lại : $(1750)_{10}$ = (1024 + 512 + 128 + 64 + 16 + 4 + 2)
= $(0000\ 0110\ 1101\ 0110)_2$

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.

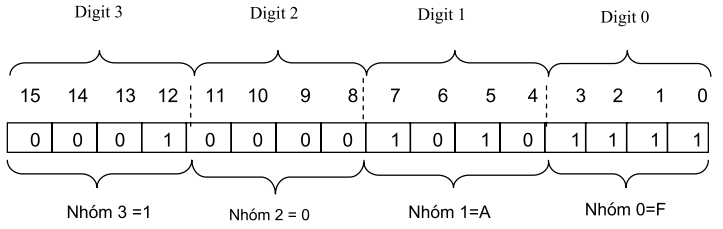


Số thập phân dưới dạng BCD :

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

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.

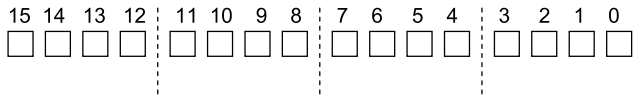


Như vậy : $0001\ \ 0000\ \ 1010\ \ 1111_2$ = $10AF_{16}$

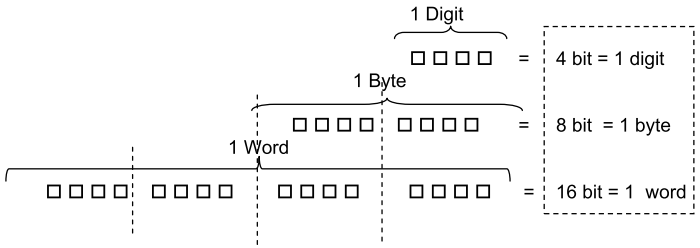
Chú ý :
- Biểu diễn số thập phân dưới dạng hexa và BCD là không hoàn toàn tương đương nhau (cho kết quả bằng dãy số nhị phân khác nhau)
- Mã BCD được dùng chủ yếu khi đổi số thập phân ra mã nhị phân dạng BCD trong khi mã hexa được dùng phổ biến khi biểu diễn dãy số nhị phân dưới dạng ngắn gọn hơn.

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).



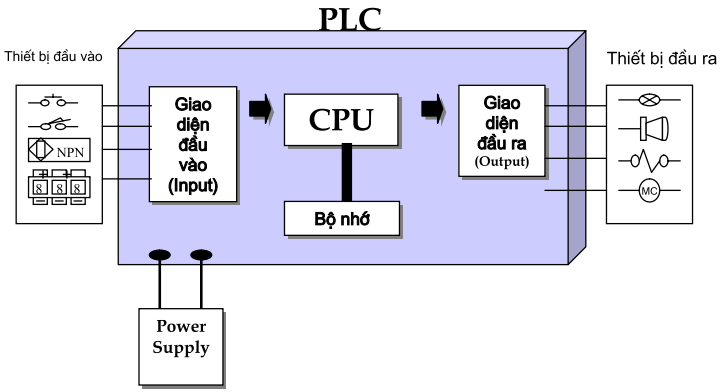
Ngoài ra để biểu diễn những số lượng lớn hơn, người ta có thêm các đơn vị sau :

- 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.

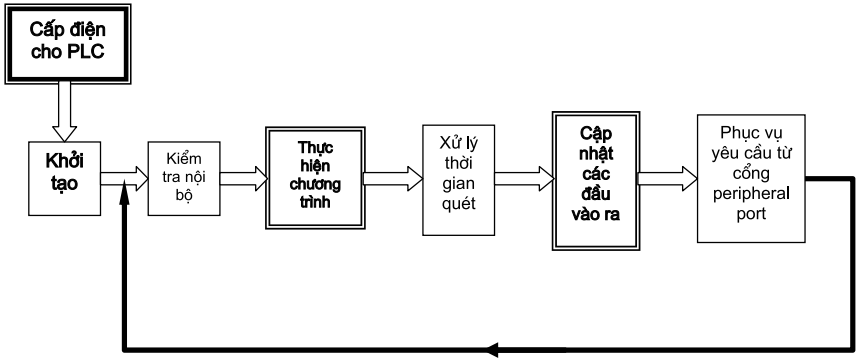
Bộ xử lý trung tâm (CPU) tuần tự thực thi các lệnh trong chương trình lưu trong bộ nhớ, xử lý các đầu vào và đưa ra kết quả kết xuất hoặc điều khiển cho phần giao diện đầu ra (output).

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 CPM1(A) và CPM2A là loại tích hợp sẵn toàn bộ các thành phần trong một bộ.

1.5 Hoạt động của PLC

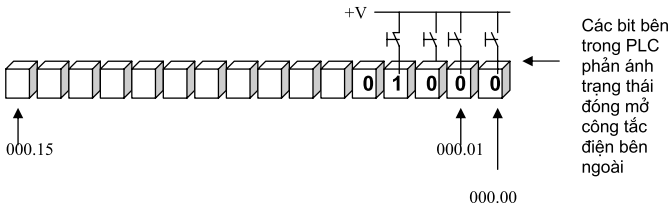
Hình 2 dưới là lưu đồ thực hiện bên trong PLC, trong đó 2 phần quan trọng nhất là **Thực hiện chương trình** và **Cập nhật đầu vào ra**. Quá trình này được thực hiện liên tục không ngừng theo một vòng kín gọi là scan hay cycle hoặc sweep. Phần thực hiện chương trình gọi là program scan chỉ bị bỏ qua 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 CPM2A, xin tham khảo catalog và tài liệu hướng dẫn sử dụng đi kèm.

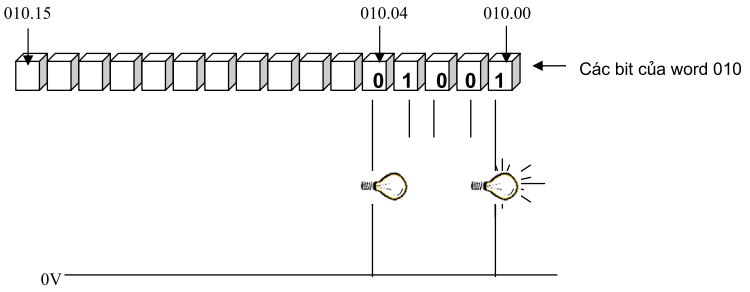
1.6 Các bit đầu vào trong PLC và các tín hiệu điện bên ngoài



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

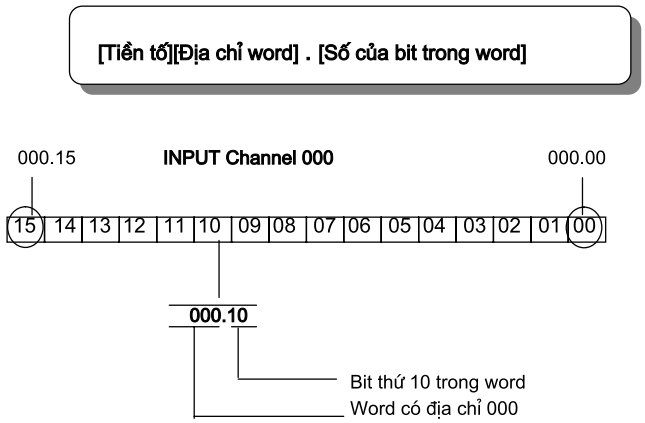


Hình 4 : Các bit đầu ra và thiết bị điện bên ngoài

Trên hình 4 là ví dụ về các bit điều khiển đầu ra của PLC. Các bit của word 010 (từ 010.00 đến 010.15) 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 CPM2A

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



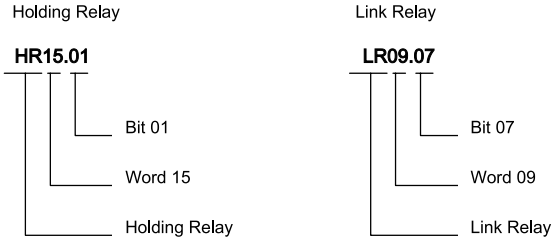
Trong đó tiền tố là ký hiệu của loại địa chỉ bộ nhớ. Ví dụ : SR cho Special Relay, LR cho Link Relay, IR cho Internal Relay,... Riêng Internal Relay là các bit vào ra I/O không cần có tiền tố IR khi tham chiếu. Special Relay cũng thường được coi là Internal Relay và không cần có tiền tố.

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 : Khi dùng Programming Console thì dấu chấm phân cách giữa địa chỉ word và bit có thể được bỏ đi; nhưng khi dùng phần mềm SYSWIN thì dấu chấm vẫn cần phải nhập vào.

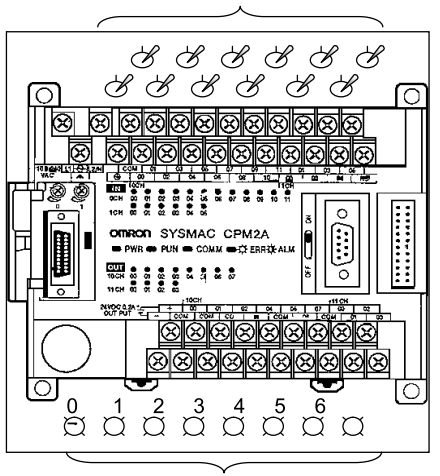
Sau đây là ví dụ về 2 trong số những bộ nhớ đặc biệt trong PLC của OMRON



PHẦN II: LÀM QUEN VỚI PLC

1.9 Giới thiệu về bộ training kit CPM2A

A) Các khoá chuyển mạch đầu vào (INPUT SWITCHES)

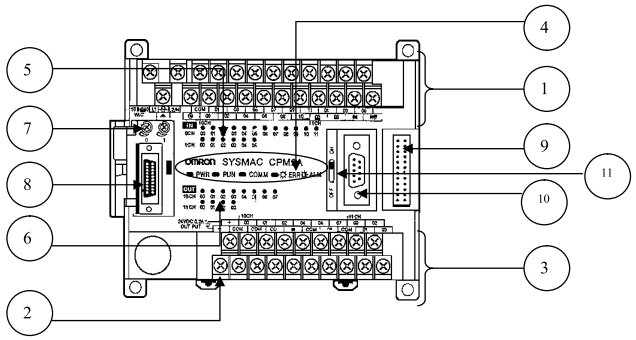


B) Các đèn chỉ thị trạng thái đầu ra (OUTPUT INDICATORS)

Hình 1 : Bộ Training CPM2A

Bộ CPM2A dành cho việc đào tạo (CPM2A Training kit) là một bộ điều khiển lập trình loại nhỏ (Micro PLC) có thêm 12 khoá chuyển mạch đầu vào để mô phỏng các đầu vào số (đánh số từ 0 đến 11) và có sẵn 8 đèn chỉ thị trạng thái đầu ra (đánh số từ 00 đến 07) được điều khiển bởi chương trình do người dùng lập (User program).

1.9.1 Các thành phần trên bộ CPM2A : Hình 2



Các thành phần chính trên bộ CPM2A trên hình :

1. **Đầu đấu dây cho**:
   - 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)
     Nối dây từ các nguồn tin hiệu ngoài vào các cực đấu dây này của PLC. Loại CPM2A-20CDR-A cung cấp 12 đầu nối vào với 1 đầu đấu chung (COMMON)

2. **Đầu nối nguồn cấp DC ra từ PLC** (DC Power Supply Output Terminal)
   Điện áp ra chuẩn là DC 24V với dòng định mức là 0,3A có thể được dùng cấp cho các đầu vào số DC.

3. **Đầu nối ra thiết bị ngoài (Output Terminal)**
   PLC loại CPM2A-20CDR-A có 8 đầu nối ra trong đó có 3 đầu COMMON

4. **Các đèn LED chỉ thị trạng thái của PLC (PC 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
ERROR/ALARM (Đỏ)	Sáng	PLC gặp lỗi nghiêm trọng (PLC ngừng chạy)
	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
COMM (Da cam)	Sáng	Dữ liệu đang được truyền qua cổng Peripheral Port
	Tắt	Không có trao đổi dữ liệu giữa PLC và thiết bị ngoài qua cổng Peripheral Port

5. **Input LED**
   Các đèn chỉ thị trạng thái đầu vào (Input Indicator)
   Đèn LED trong nhóm này sẽ sáng khi đầu vào tương ứng lên ON

   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.

6. **Output LED (Output Indicator)**: Các đèn chỉ thị trạng thái đầu ra. Các đèn LED này sẽ sáng khi rơ le tương ứng được bật.

7. **Analog Setting Controls**
   PLC loại CPM2A có 2 bộ chỉnh giá trị thanh ghi bên trong PLC đánh số 0 và 1. Mỗi khi núm điều chỉnh được vặn, giá trị của thanh ghi tương ứng được thay đổi trong khoảng giá trị từ 000 đến 200 (theo mã BCD). Các thanh ghi trong PLC tương ứng với 2 bộ chỉnh này là IR250 và IR251. Nếu gán địa chỉ tham chiếu của timer hoặc counter với các địa chỉ này ta có thể điều chỉnh giá trị của chúng bằng tay không cần đến phần mềm hỗ trợ.

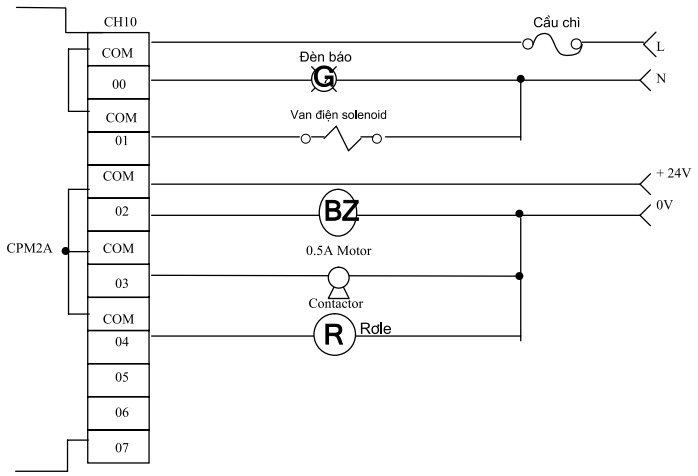
8. **Peripheral Port**
   Dùng để nối PLC với thiết bị ngoại vi, bộ chuyển đổi RS-232 hay RS-485 hoặc bộ lập trình cầm tay (Programming Console)

9. **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.

10. **Cổng RS-232C**
    dùng giao tiếp với các thiết bị khác như bộ xử lý tín hiệu số, bộ điều khiển nhiệt độ,...

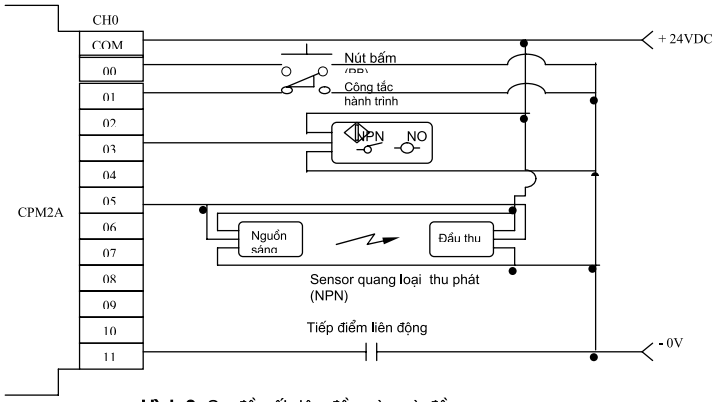
11. **Communications Switch:**
    công tắc chuyển dùng đặt cấu hình cho truyền tin.

1.9.2 Ví dụ về đấu dây CPM2A



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 CPM2A.

Bảng 2 Địa chỉ bộ nhớ vào ra của các loại PLC họ CPM2A (20,30,40,60 I/O)

Số lượng đầu vào ra trên module	Đầu nối trên module CPU		Đầu vào ra trên module mở rộng CPM1A-8E□□ CPM1A-20E□□		Model N⁰
CPU	Input	Output	Input	Output	
20	12 đầu: 00000 đến 00011	8 đầu: (rơle/transistor) 01000 đến 01007 (rơle/transistor)	m đầu: (m=8 hoặc 12 tuỳ module) Word (k+1), bit 00 đến m-1	8 đầu: (tuỳ module) Word (i+1), bit 00 đến 07	CPM2A-20CDR/T-A
					CPM2A-20CDR/T-D
30	18 đầu: 00000 đến 00011 00100 đến 00105	12 đầu: 01000 đến 01007 01100 đến 01103			CPM2A-30CDR-A
					CPM2A-30CDR-D
40	24 đầu 00000 đến 00011 và 00100 đến 00111	16 đầu 01000 đến 01007 Va2 01100 đến 01107			CPM2A-40CD□
60	36 đầu 00000 đến 00011 và 00100 đến 00111 00200 đến 00211	24 đầu 01000 đến 01007 và 01100 đến 01107 và 01200 đến 01207			CPM2A-60CD□

Trong đó:
k= word input cuối của CPU hoặc word đã được phân cho Expansion Unit kế trước nếu như Expansion Unit này đã nối.
i= word output cuối của CPU hoặc word đã được phân cho Expansion Unit kế trước nếu như Expansion Unit này đã nối.

Ví dụ. Với bộ CPM2A-30CDR-A với 30 đầu vào/ra thì:
- Trên CPU Unit: Input chiếm các word 000 và 001
  Output chiếm các word 010 và 011
- Nếu nối thêm module mở rộng CPM2A-20EDR (12 vào/8 ra) thì :
  Input chiếm word 002, các bit từ 00 đến 11
  Output chiếm word 012, các bit từ 00 đến 07
- Nếu nối thêm tiếp module mở rộng CPM2A-20EDT (12 vào/8 ra) thì:
  - Input chiếm word 003, các bit từ 00 đến 11
  - Output chiếm word 013, các bit từ 00 đến 07
- Nếu nối thêm tiếp module mở rộng CPM2A-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

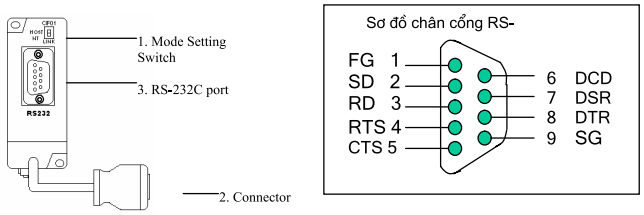
Về các module khác, xin tham khảo tài liệu đi kèm của các module này, catalog hoặc cuốn "Programming Manual".

1.10 Nối ghép giữa PLC và thiết bị ngoại vi :

Để PLC có thể giao tiếp được với các thiết bị ngoại vi qua cổng Peripheral Port, cần có một bộ chuyển đổi RS-232 hoặc RS-422 bên ngoài.

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

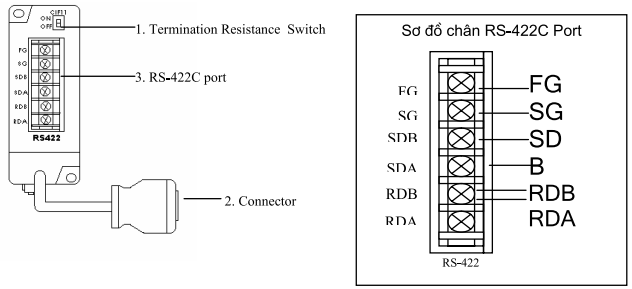
1.10.1 RS-232C Adapter (CPM1-CIF01)



Hình 4 : RS-232C Adapter

1. *Mode Setting Switch*: Đặt khoá Mode về vị trí ***Host*** khi muốn dùng Host Link System để nối với máy tính. Đặt về vị trí "***NT***" khi nối với một Programmable Terminal hoặc một PLC khác dùng giao thức "1:1" NT link.
2. *Connector* : Nối với cổng Peripheral Port của CPU PLC.
3. *RS-232C Port* : Nối với cáp RS-232C từ thiết bị khác như máy tính, thiết bị ngoại vi hay Programmable Terminal.

1.10.2 RS-422 Adapter (CPM1-CIF11) (Link Adapter)



Hình 5: RS-422 Adapter

1. *Terminator Resistance Switch* : Đặt khoá này về vị trí "ON" (vị trí trên) cho các Link Adapter ở cả 2 đầu của hệ thống giao tiếp dùng Host Link và cho RS-422 Adapter.
2. *Connector* : Nối với cổng Peripheral Port của CPU PLC.
3. *RS-422 Port* : Nối với mạng Host Link dùng chuẩn RS-422C.

1.11 Các loại module mở rộng (Expansion I/O Unit)

Bảng 3: **Các loại module mở rộng của họ CPM2A**

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

1.12 Các tính năng chính của bộ CPM2A

1.12.1 Module CPM2A chính cung cấp 4 loại với số lượng I/O khác nhau : 20, 30, 40 và 60 I/O (xem *bảng 2*). Tất cả đều có sẵn cổng RS-232C.

1.12.2 Có thể lắp thêm tối đa là 3 module mở rộng (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 CPM2A 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 dạng ngôn ngữ bậc thang bằng phần mềm chạy trong DOS với SYSMAC Support Software: (SSS) hoặc trong Windows với SYSWIN, hoặc dạng dòng lệnh dùng bộ lập trình cầm tay.

1.12.5 Có 2 chiết áp chỉnh độ lớn thanh ghi bên trong PLC (Analog Volume Setting) với khoảng thay đổi giá trị từ 0-200 (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 . . . 5 KHz
- UP/DOWN mode . . . 2,5 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).

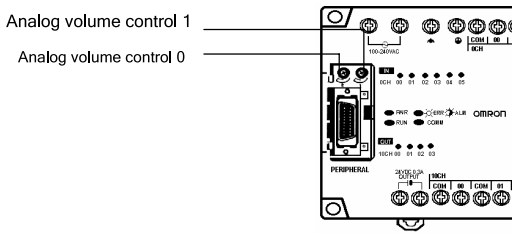
1.12.8 Có đầu vào tốc độ cao để phát hiện các tín hiệu với độ rộng xung ngắn (tới 0,2msec) không phụ thuộc vào thời gian quét chương trình.

1.12.9 Truyền tin theo chuẩn Host Link/NT Link hoặc 1:1 Data Link qua cổng Peripheral Port hoặc cổng RS-232C có sẵn trên CPU Unit.

1.13 Analog Setting Function

Bộ CPM2A có 2 đầu vào chiết áp để chỉnh giá trị thanh ghi bên trong PLC (Analog Setting Function) với độ phân giải 8 bit và khoảng giá trị thay đổi từ 0-200 (BCD).

Analog Volume Control 0 → SR250
Analog Volume Control 1 → SR251



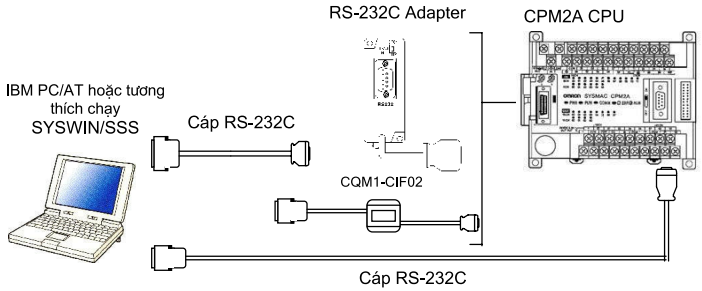
Hình 6: Chiết áp chỉnh thanh ghi bên trong PLC

1.14 Giao tiếp truyền tin (Communications)

1.14.1 Giao tiếp dùng Host Link

Giao tiếp dùng Host Link cho phép tới 32 bộ PLC có thể được điều khiển bởi 1 máy tính chủ (Host Computer). Host Link có thể dùng RS-232C hoặc RS-422C Adapter. 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 Peripheral Port.

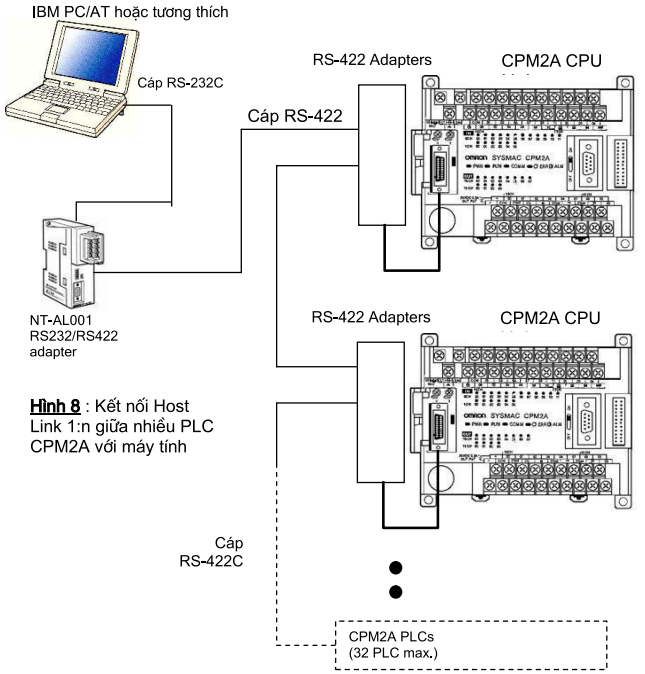
➢ **Kết nối 1:1**



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.



Hình 8 : Kết nối Host Link 1:n giữa nhiều PLC CPM2A với máy tính

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

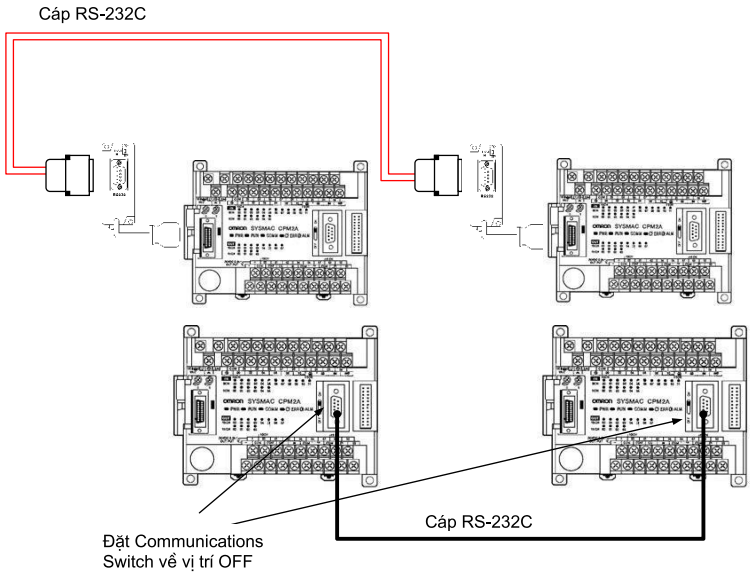
Bảng 4 Các loại cáp nối và adapter :

Loại	Công dụng	Mã
RS-232C Adapter	Chuyển đổi giữa chuẩn RS-232 và chuẩn điện của Peripheral Port	CPM1-CIF01
RS-422 Adapter		CPM1-CIF11
RS-232C Adapter + Cáp nối	Bộ chuyển đổi có sẵn cáp để nối với máy tính IBM PC/AT hoặc tương thích (Chiều dài cáp : 3,3m)	CQM1-CIF02
Link Adapter	Chuyển đổi giữa 2 chuẩn điện RS-232C và RS-422	NT-AL001

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 vùng thanh ghi LR giữa 1 bộ CPM2A với 1 bộ PLC loại CPM1(A), CPM2A, CQM1, C200HS, C200HE/G/X hay SRM1. Để thực hiện liên kết cần có cáp RS-232C và bộ chuyển đổi RS-232C Adapter (nếu dùng cổng Peripheral). 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
RS-232C Adapter	Chuyển đổi giữa chuẩn RS-232 và chuẩn điện của Peripheral Port (nếu dùng cổng Peripheral Port)	CPM2A-CIF01
Cáp nối	Để nối giữa các PLC với nhau (chuẩn RS-232C) (max. 15m)	Tự tạo hoặc mua



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

➢ **Ví dụ về liên kết 1:1 giữa 2 bộ CPM2A**

Trong mỗi bộ CPM2A, có một vùng bộ nhớ đặc biệt gọi là "Link Relay" hay Link Bits (được ký hiệu với tiền tố LR) 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 LR 00 - LR 15 (tổng cộng 256 bit). Khi kết nối, một PLC phải được đặt là **master**, còn PLC kia là **slave**.

Bước 1: **Đặt thông số trong PLC**

Mỗi bộ PLC cần có 1 bộ chuyển đổi RS-232C và cáp nối giữa 2 PLC với nhau. Khoá chuyển (DIP switch) trên mỗi bộ RS-232C Adapter phải đặt về vị trí "NT". Khi 2 PLC đang trao đổi dữ liệu với nhau, đèn LED "COMM" trên cả 2 PLC sẽ nhấp nháy để biểu thị sự hoạt động của PLC.

Để đặt chế độ kết nối truyền tin giữa 2 PLC, thanh ghi DM6645 trong *mỗi* bộ CPM2A phải được đặt (setting) như bảng dưới đây trong đó có 1 bộ là Master (DM6645 = 3000), còn bộ kia là Slave (DM6645=2000).

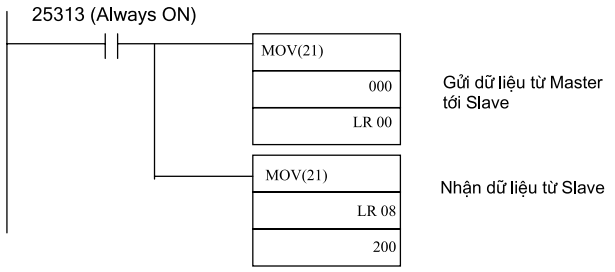
Word	Bit	Chức năng	Setting (Master)	Setting (Slave)
DM 6645	00 đến 03	Cấu hình cổng : 00: Chuẩn (1 start bit, 7-bit data, even parity, 2 stop bits, 9,600 bps) 01: Setting lưu ở DM 6651	00 (Tuỳ ý)	00 (Tuỳ ý)
	04 đến 07	Chế độ kiểm tra CTS	0	0
	08 đến 11	Link Area cho 1:1 PC link qua cổng peripheral port 0: LR 00 đến LR 15	0	0 (Tuỳ ý)
	12 đến 15	Chế độ truyền tin 0: Host Link; 2:1:1 PC Link (Slave); 3:1:1 PC Link (Master); 4: NT Link	3	2

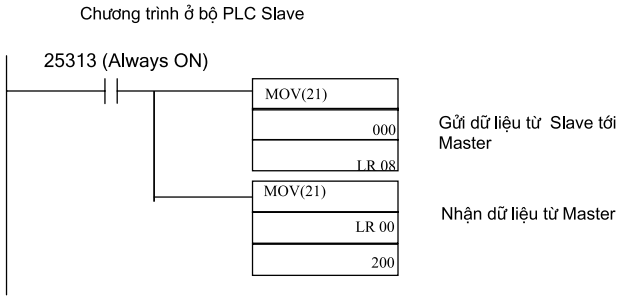
Bước 2 : **Viết chương trình truyền và nhận dữ liệu**

Mỗi bộ CPM2A 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 :

Ở bộ CPM2A Master

Chương trình ở bộ PLC Master





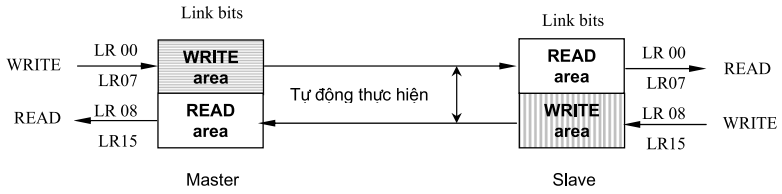
Ở bộ CPM2A Slave

Hoạt động của hệ thống Data Link

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

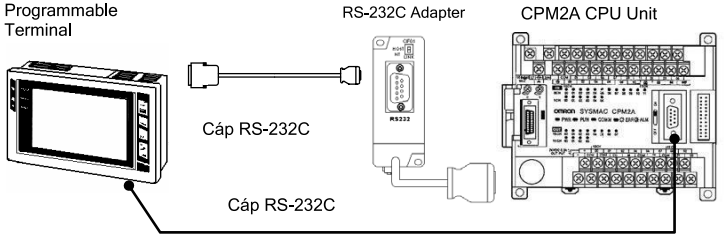
Ở bộ PLC master : Dữ liệu từ thanh ghi [IR] 000 được chuyển (bằng lệnh MOV) vào thanh ghi LR00. Sau đó, dữ liệu ở LR00 của bộ Master được tự động truyền sang thanh ghi LR00 ở PLC slave đồng thời dữ liệu từ thanh ghi LR08 (nhận được từ PLC slave) được chuyển (copy) vào thanh ghi 200 của PLC master.

Ở bộ PLC Slave : Dữ liệu từ thanh ghi [IR] 000 được chuyển vào thanh ghi LR08. Sau đó, dữ liệu ở LR08 của bộ Slave được tự động truyền sang thanh ghi LR00 ở PLC Master đồng thời dữ liệu từ thanh ghi LR00 (nhận được từ PLC master) được chuyển vào thanh ghi 200 của PLC slave.



1.14.3 Truyền tin dùng NT Link

NT Link cung cấp phương tiện trao đổi dữ liệu nhanh bằng phương thức truy cập trực tiếp giữa bộ CPM2A với Programmable Terminal-PT (dùng NT Link Interface) và sử dụng RS-232C Adapter hoặc trực tiếp với cổng RS-232C.



Hình 10 : Thiết bị cần dùng cho NT Link

Bộ	Công dụng	Model N^0
RS-232C Adapter	Chuyển đổi sang dạng chuẩn của Peripheral Port	CPM1-CIF01
Cáp nối RS-232C	Nối giữa bộ chuyển đổi và cổng của PT	Tự tạo hoặc mua

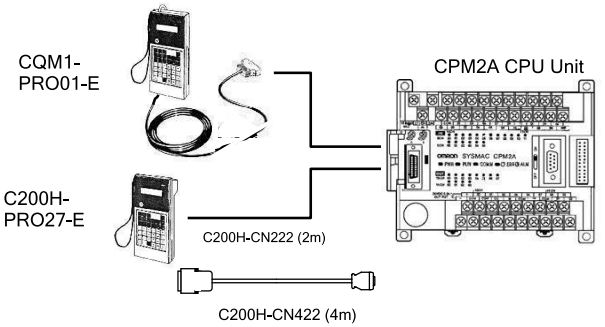
- CPM1-CIF01 khi dùng với NT Link qua cổng Peripheral Port phải được đặt về vị trí "NT" (vị trí dưới).

1.15 Peripheral Device

CPM2A có thể nối với các thiết bị ngoại vi khác bao gồm Programming Console hoặc Personal Computer (PC) có chạy phần mềm Sysmac Support Software (SSS), SYSWIN hay Sysmac CPT.

1.15.1 Bộ lập trình cầm tay (Programming Console) :

Hình 11 : CPM2A và Programming console : CQM1-PRO01-E & C200H-PRO27-E



Các Programming Console sau có thể được dùng để lập trình cho CPM2A: CQM1-PRO01-E và C200H-PRO27-E

Bảng 5 Các loại Programming Console và cáp

Bộ		Model N⁰
CQM1-Series Programming Console (cáp nối kèm sẵn)		CQM1-PRO01-E
C200H-Series Programming Console		C200H-PRO27-E
Cáp nối cho C200H-Series Programming Console	Chiều dài cáp : 2 m	C200H-CN222
	Chiều dài cáp : 4 m	C200H-CN422

1.15.2 Phần mềm lập trình cho PLC:

Phần mềm lập trình cho PLC có thể được cài đặt trên máy tính IBM PC/AT hoặc tương thích với 2 loại :

- Loại chạy trên DOS : **SYSMAC Support Software (SSS)**
- Loại chạy trong Windows **: SYSWIN V3.3/3.4 hoặc CX-Programmer**

Bảng 6 Các phụ kiện cho kết nối PLC - phần mềm lập trình (vd: SYSMAC Support Software)

Tên	Công dụng	Model N⁰
RS-232C Adapter	Để chuyển đổi sang chuẩn của cổng Peripheral	CPM1-CIF01
RS-232C Adapter + Cáp nối	Bộ chuyển đổi có sẵn cáp để nối với máy tính (Chiều dài : 3,3 m)	CQM1-CIF02
Ladder Support Software (chạy trong Windows)	Cho máy IBM AT hoặc tương thích (3.5" Disks, 2HD)	SYSWIN V3.3/3.4

1.16 Các vùng nhớ trong CPM2A

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

Vùng nhớ		Words	Bits	Chức năng
IR area[1]	Input area	IR000 đến IR009 (10 words)	IR00000 tới IR 00915 (160 bits)	Các bit này có thể được gán cho các đầu dây vào ra I/O. Tiền tố IR thường được bỏ đi
	Output area	IR010 tới IR019 (10 words)	IR01000 tới IR 01915 (160 bits)	
	Work area	IR20 tới IR49 (30 words) IR200 tới IR227 (28 words)	IR 02000 tới IR 04915 IR20000 tới IR 22715 (512 bits)	Work bit có thể được sử dụng tuỳ ý trong chương trình
SR area		SR228 tới SR255 (28 words)	SR22800 tới SR 25515	Các bit này phục vụ cho các chức năng riêng biệt như cờ báo và các bit điều khiển.
TR area		---	TR0 tới TR7 (8 bits)	Các bit này lưu dữ liệu và lưu trạng thái ON/OFF tạm thời tại các nhánh rẽ chương trình.
HR area[2]		HR00 tới HR19 (20 words)	HR0000 tới HR 1915 (320 bits)	Các bit này lưu dữ liệu và lưu lại trạng thái ON/OFF của chúng khi ngắt điện.

AR area[2]		AR00 tới AR23 (24 words)	AR0000 tới AR 2315 (256 bits)	Các bit này phục vụ cho các chức năng riêng biệt như cờ báo và các bit điều khiển.
LR area[1]		LR00 tới LR15 (16 words)	LR0000 tới LR 1515 (256 bits)	Dùng cho kết nối 1:1 với 1 PLC khác.
Timer/Counter area[2]		TC000 tới TC255 (timer/counter numbers)[3]		Các số này có thể được dùng cho cả timers và counters.
DM area	Read/Write[2]	DM0000 tới DM2047 (2,048 words)	---	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]	DM2000 tới DM 2021 (22 words)	---	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]	DM6144 tới DM 6599 (456 words)	---	Chương trình không thể ghi đè lên các word này
	PC Setup[4]	DM6600 tới DM 6655 (56 words)	---	Dùng lưu các thông số khác nhau điều khiển hoạt động của PLC.

Ghi chú :

1. Các bit IR và LR khi không được dùng cho các chức năng đã định của chúng có thể được dùng như bit chương trình (work bit).
2. Nội dung của các thanh ghi HR, LR, counter, và vùng bộ nhớ DM đọc/ghi được nuôi bằng tụ. Ở nhiệt độ 25⁰C, tụ có thể lưu nội dung bộ nhớ trong vòng 20 ngày.
3. Khi truy cập giá trị hiện hành (PV) của timer và counter, các số của timer và counter (ví dụ CNT001, TIM005) 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.

CHƯƠNG 3

PHẦN MỀM CX-PROGRAMMER

Vài nét về bộ phần mềm CX-Automation Suite

CX-Automation Suite 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.

	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.
	PLC Reporter 32	PLC Reporter 32 cho phép người sử dụng đọc và ghi dữ liệu từ PLC bằng Microsoft Excel mà không cần phải lập trình
	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-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-Process Monitor Plus	CX-Process Monitor Plus là công cụ thiết yếu để theo dõi các quá trình của từng khối chức năng, thay đổi các tham số và cấu hình báo động. Công cụ này kết hợp với phần mềm CX-Process Tools để tạo nên 1 bộ công cụ mạnh cho theo dõi và điều khiển quá trình
	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 tin của họ PLC CS1/CJ1 & các họ PLC khác. Sau đó việc thực hiện truyền tin 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-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 2 phiên bản chính:
- Bản Junior: 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 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.

Phần tiếp theo xin giới thiệu từng bước về 1 số các thao tác cơ bản với CX-Programmer.

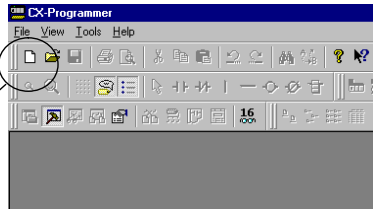
Các ký hiệu quy ước dùng trong tài liệu:

Chỉ thao tác bấm nút trái chuột

Chỉ thao tác bấm đúp nút trái chuột

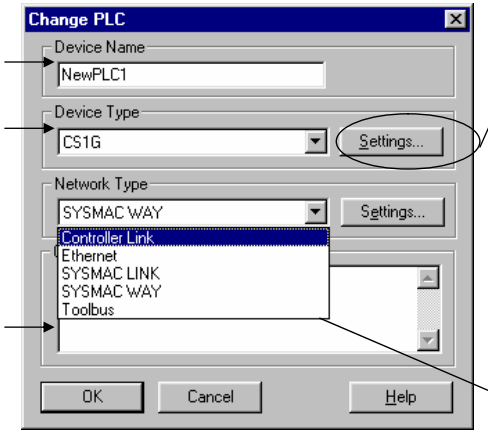
Chỉ thao tác bấm nút phải chuột

Tạo 1 project mới



Bấm nút New để tạo project mới

Bấm để chọn loại CPU trong series



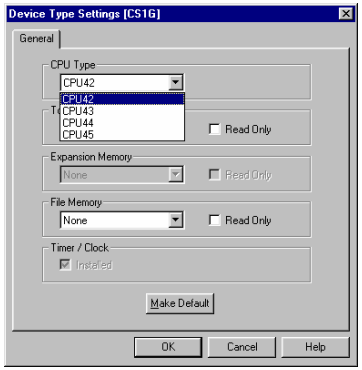
Đặt tên cho PLC

Chọn đúng series PLC

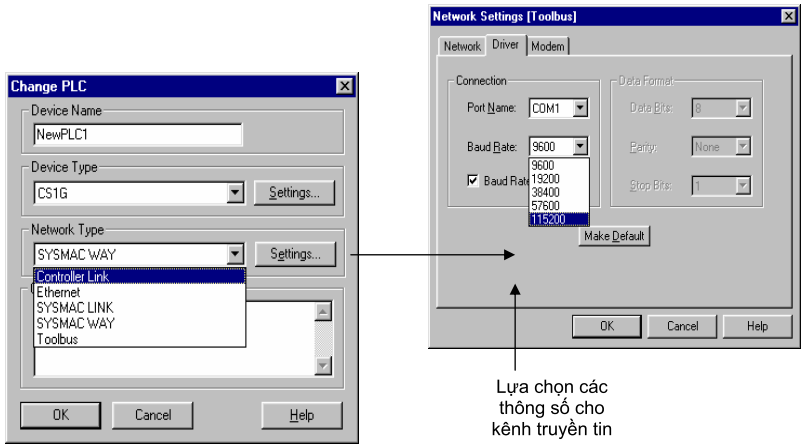
Mô tả dự án

Chọn kênh truyền tin với PLC

Chọn loại CPU

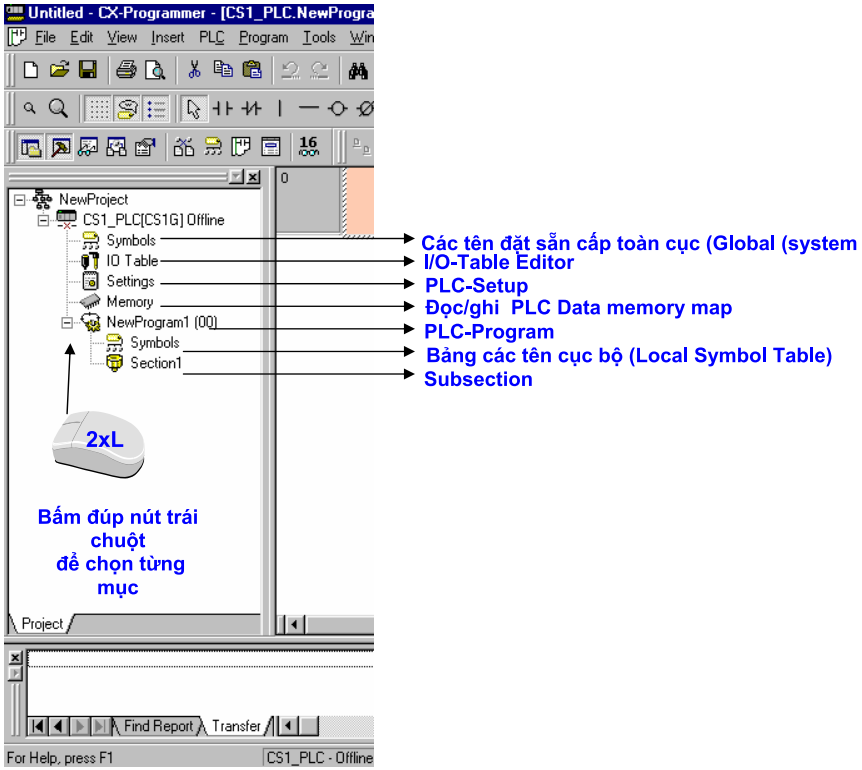


Chọn kênh truyền tin



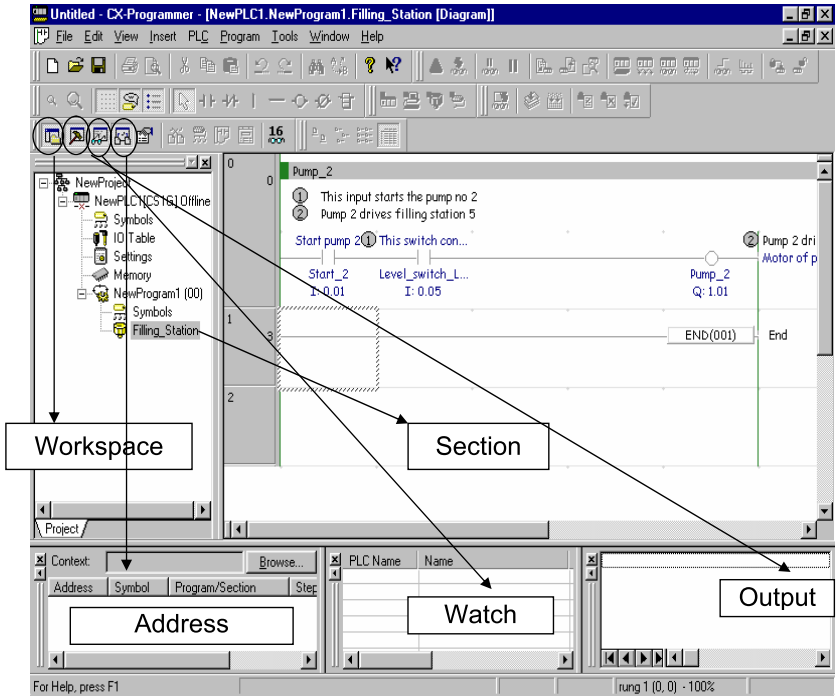
Các thông số này thường là không cần thay đổi vì các thông số mặc định đã được đặt sẵn phù hợp với loại PLC đang dùng. Trường hợp CX-Programmer không thể kết nối với PLC, hãy kiểm tra lại thông số này.

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



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

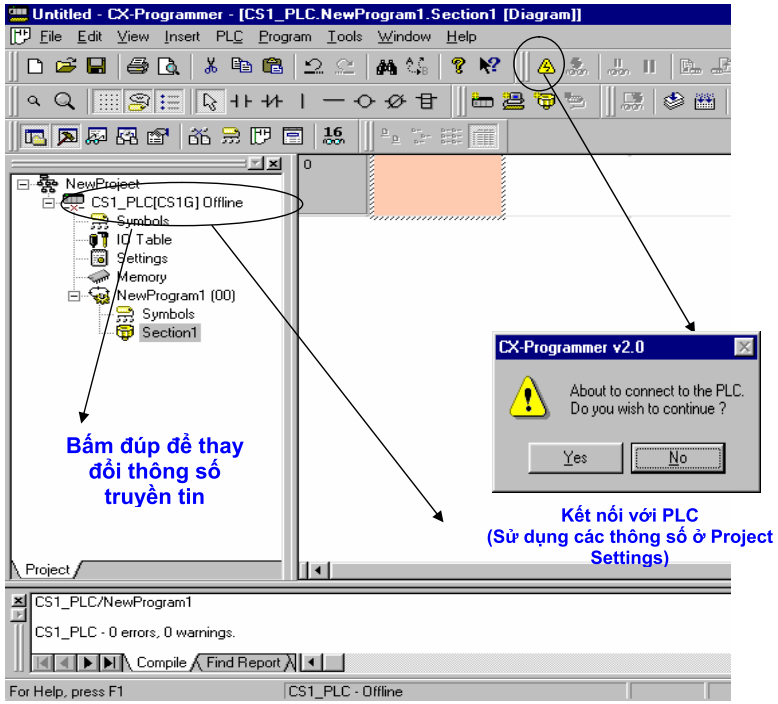
Trong quá trình làm việc với **CX-Programmer**, người sử dụng có thể bật hoặc tắt các cửa sổ phụ. Các cửa sổ này hiển thị các thông tin có liên quan đến các đối tượng & công việc đang được thực thi.



- 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.

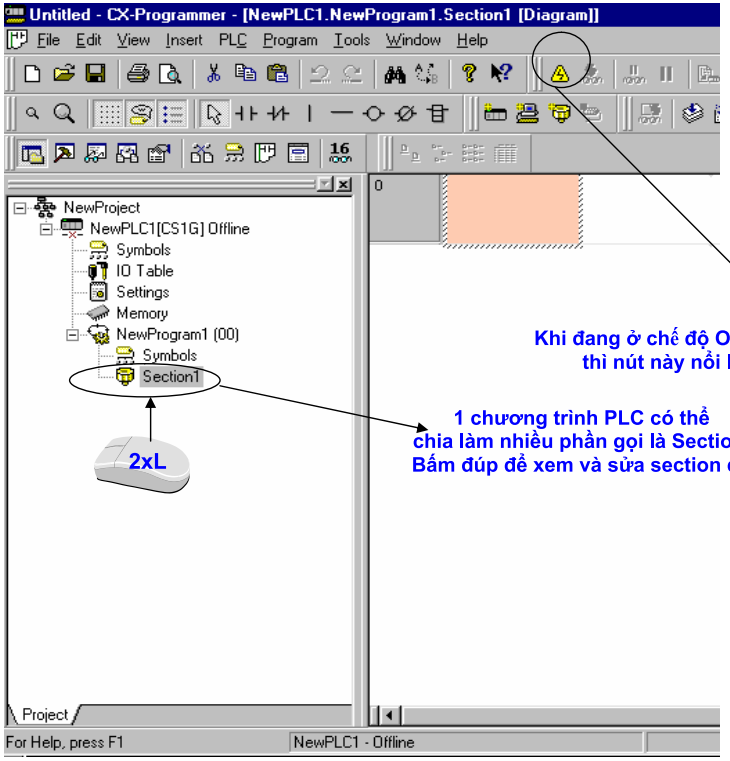
Kiểm tra kết nối (Communication) với PLC

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.

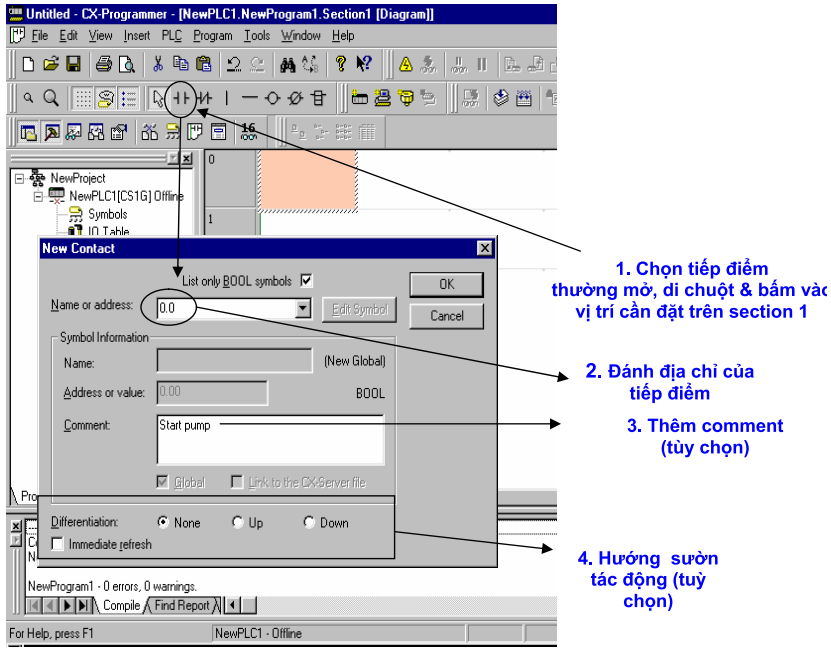


Bấm lại vào nút Work Online sẽ chuyển sang chế độ Offline để có thể sửa chương trình

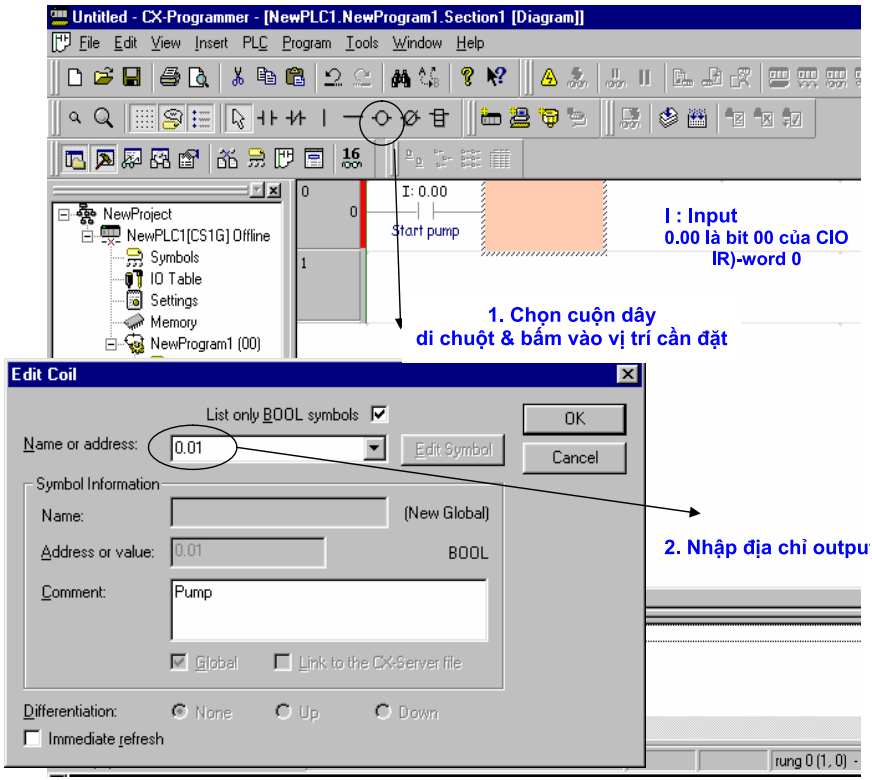
Bấm đúp vào Section1 để hiển thị cửa sổ sửa chương trình bên phải



Thêm tiếp điểm

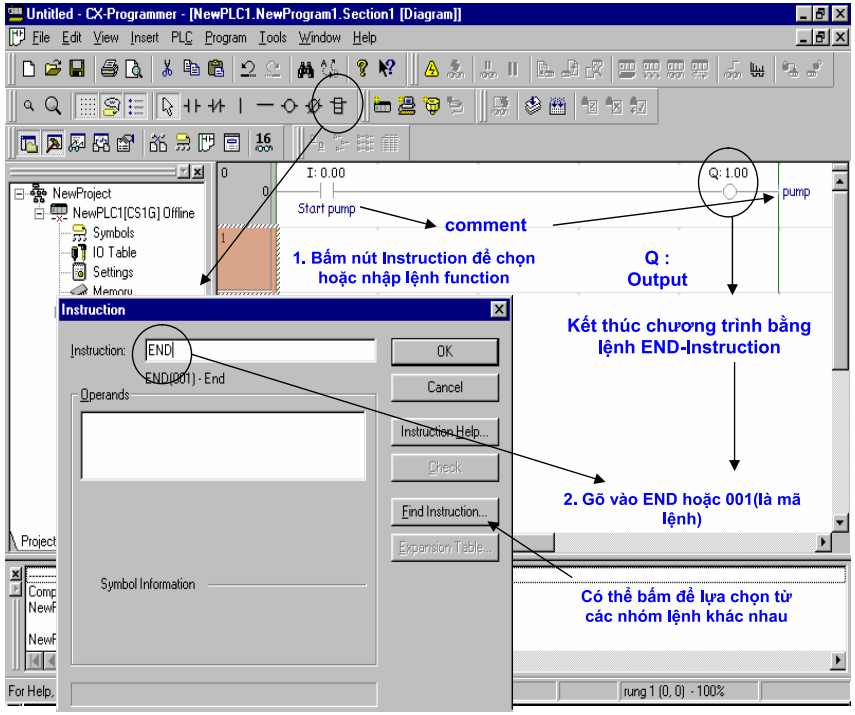


Thêm cuộn dây



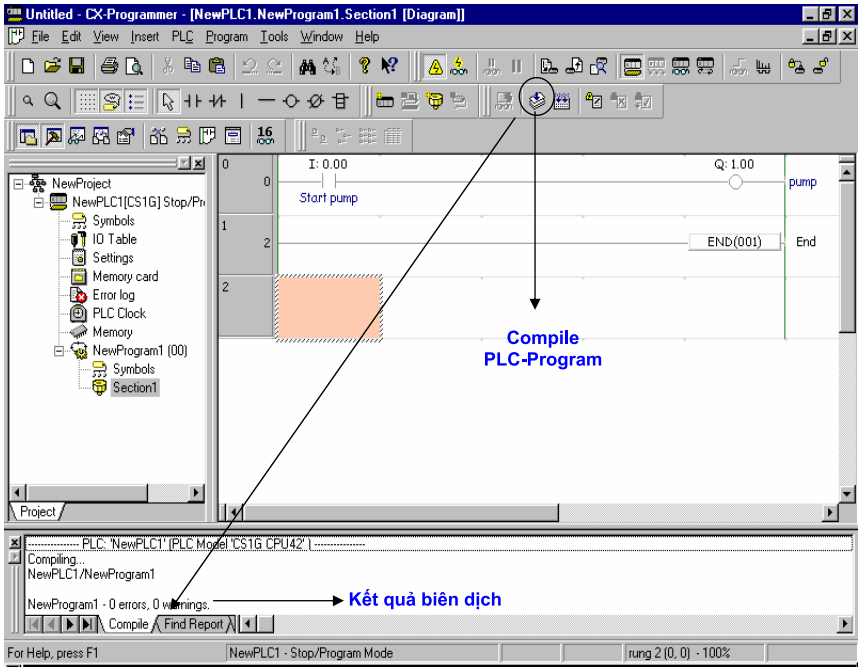
Thêm function

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.



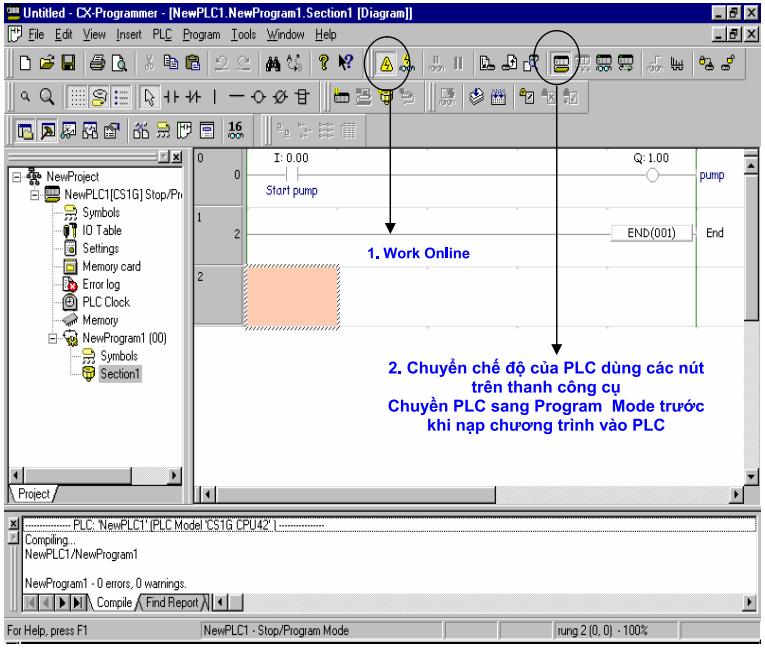
Kiểm tra & biên dịch chương trình

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.



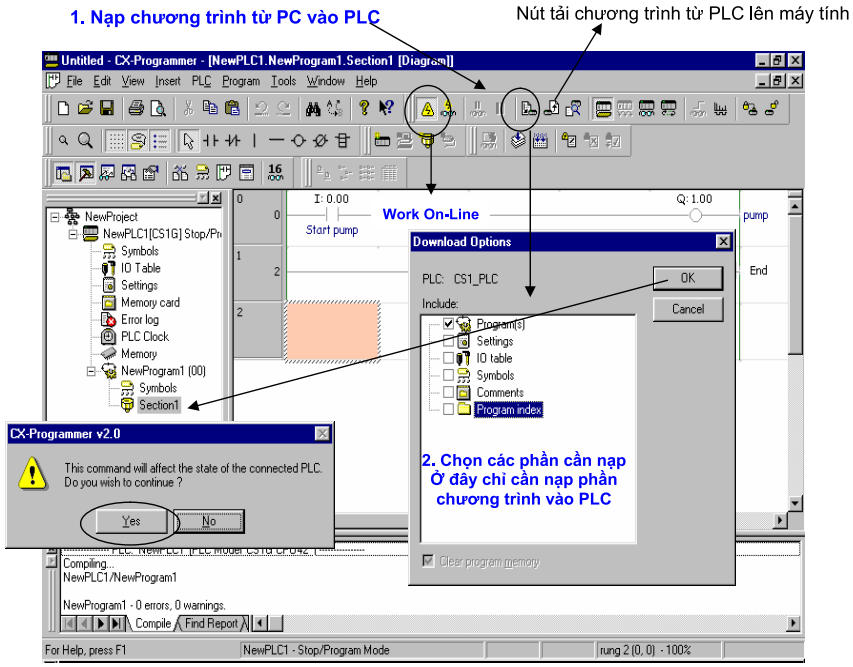
Bước tiếp theo chúng ta sẽ nạp chương trình đã viết vừa qua vào PLC. Về nguyên tắc, PLC cần chuyển sang Program Mode trước khi cho phép thay đổi nội dung chương trình PLC. Tuy vậy, ta có thể nạp chương trình vào PLC kể cả khi đang ở bất kỳ chế độ nào nhờ có các tính năng của CX-Programmer trợ giúp.

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.

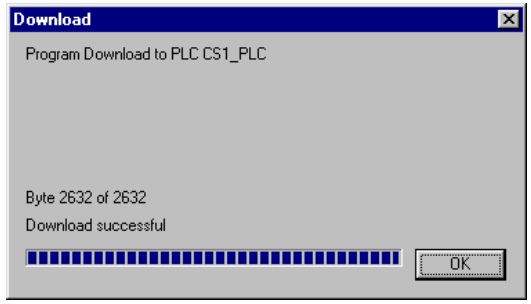


Khi đang online với PLC, các nút này cũng trực tiếp phản ánh chế độ làm việc hiện hành của PLC.

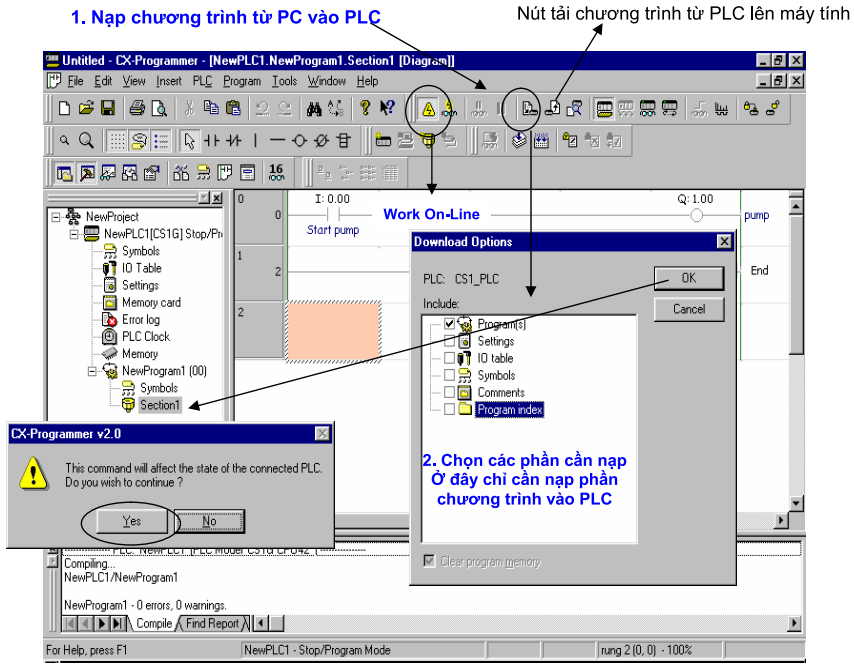
Nạp (Download) chương trình vào PLC



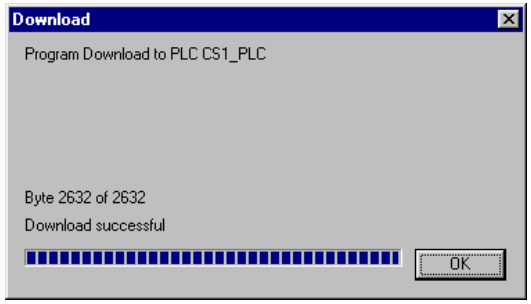
Việc nạp chương tình vào PLC cũng sẽ xóa nội dung hiện đang có trong PLC. Vì thế cần thận trọng xác nhận việc này trước khi tiếp tục.



Phần mềm CX- Programmer Chương III
Nạp (Download) chương trình vào PLC

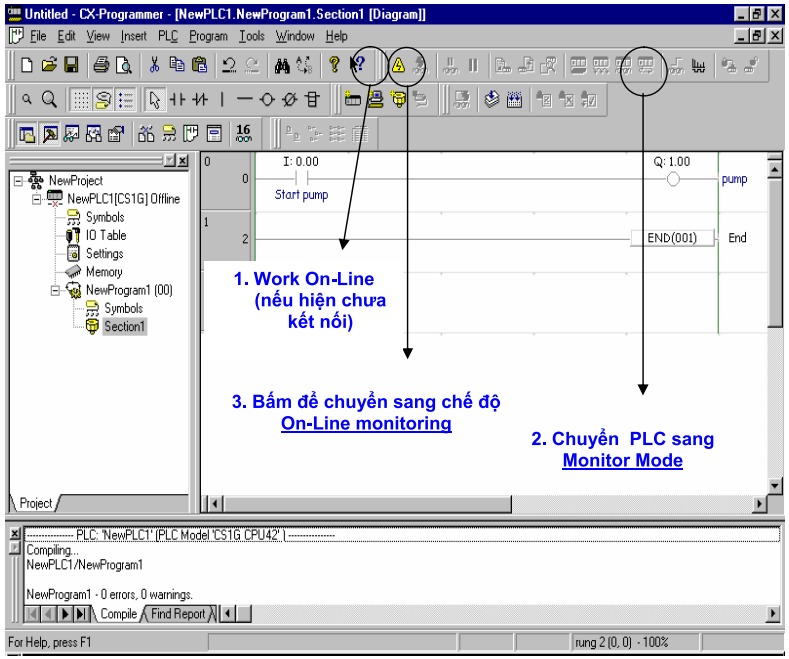


Việc nạp chương tình vào PLC cũng sẽ xóa nội dung hiện đang có trong PLC. Vì thế cần thận trọng xác nhận việc này trước khi tiếp tục.



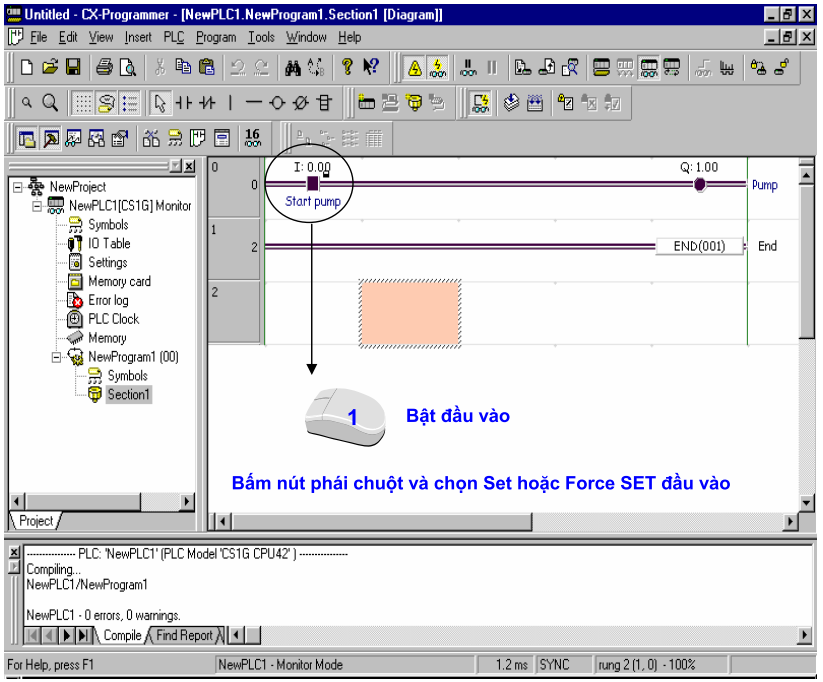
Chuyển PLC sang chế độ Monitor mode

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.



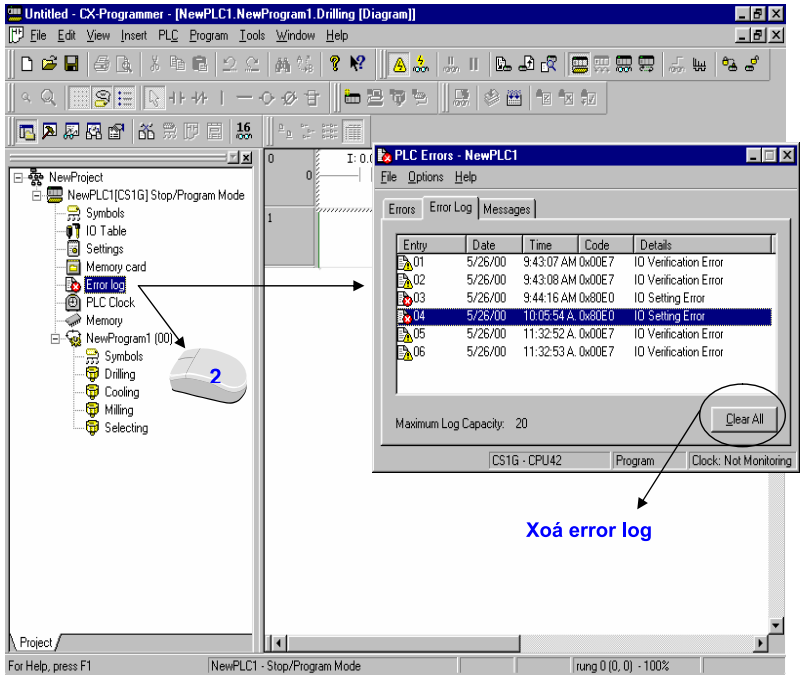
Thử chương trình

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.

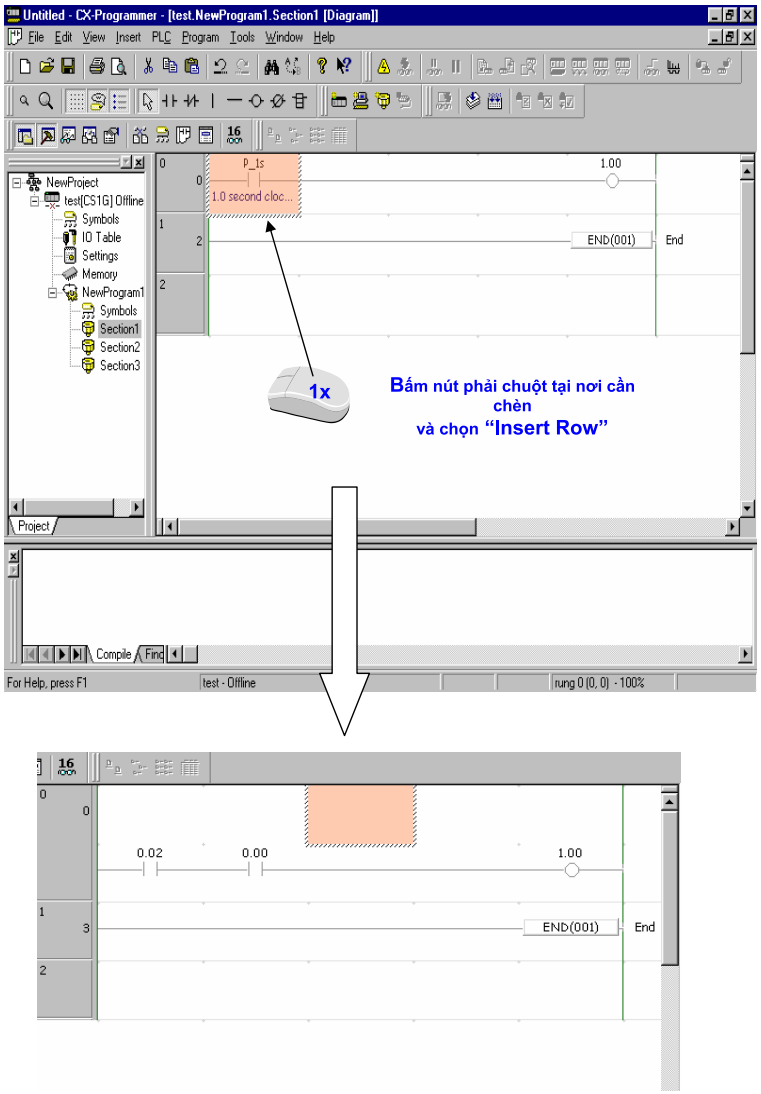


Kiểm tra bản ghi lỗi trong PLC

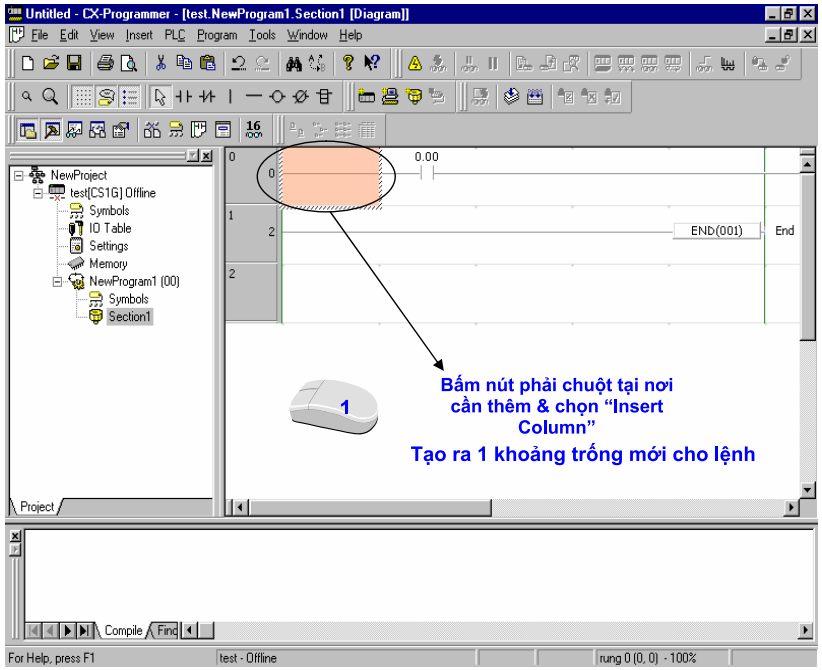
Khi đang online có thể kiểm tra và xóa các lỗi đang có trong PLC bằng cách bấm đúp vào Error Log.



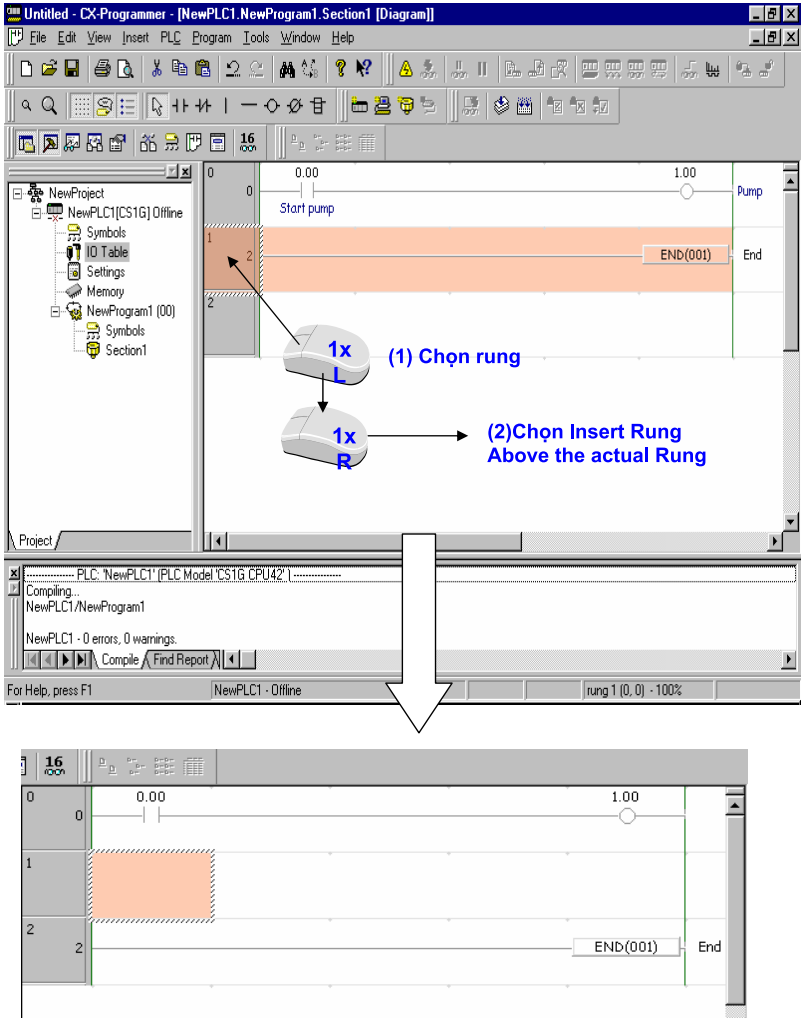
Thêm hàng vào Rung



Thêm cột vào Rung



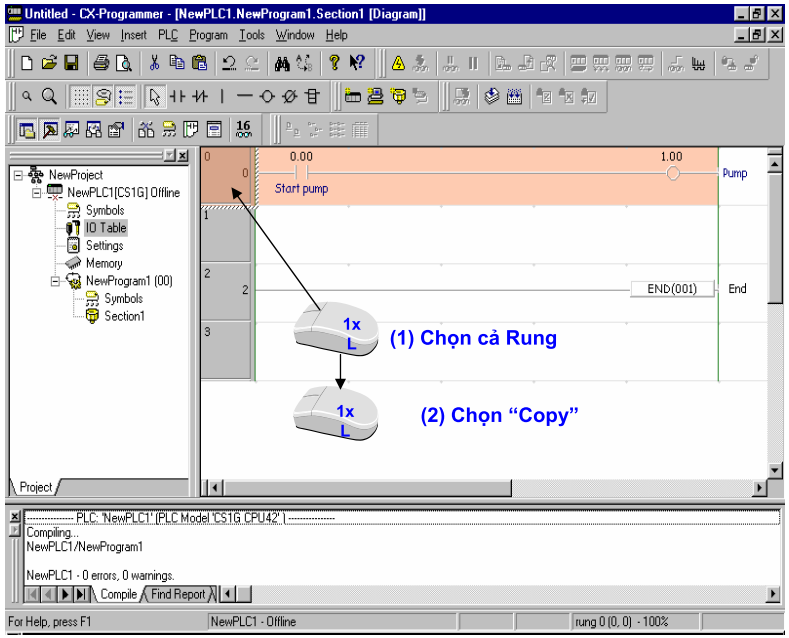
Chèn thêm 1 rung



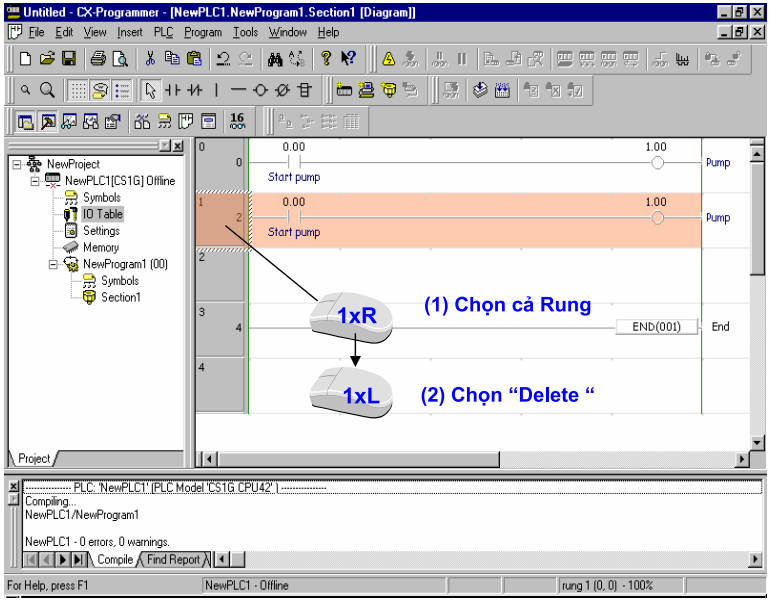
Các thao tác Copy & Paste

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.

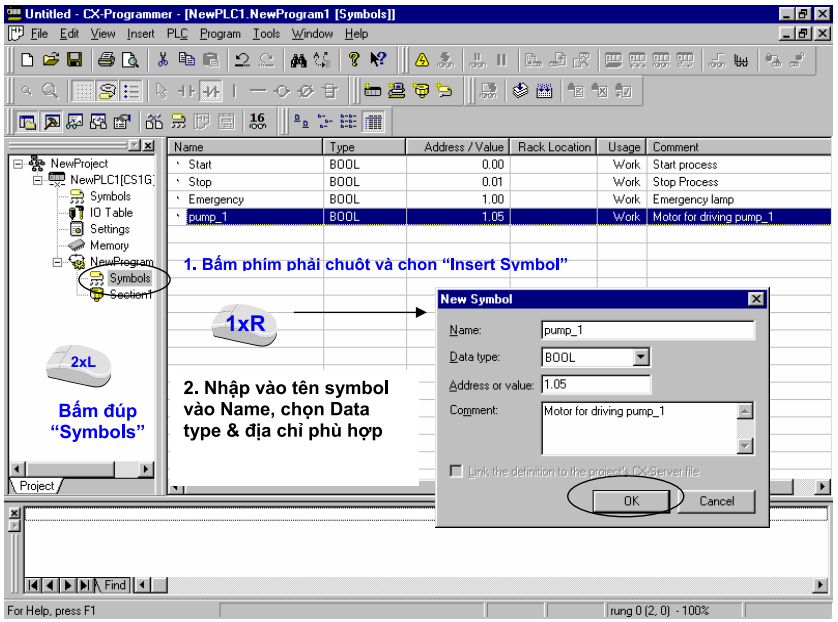
Dưới đây là ví dụ thao tác Copy cả 1 rung rồi paste vào 1 chỗ khác.



Xoá Rung

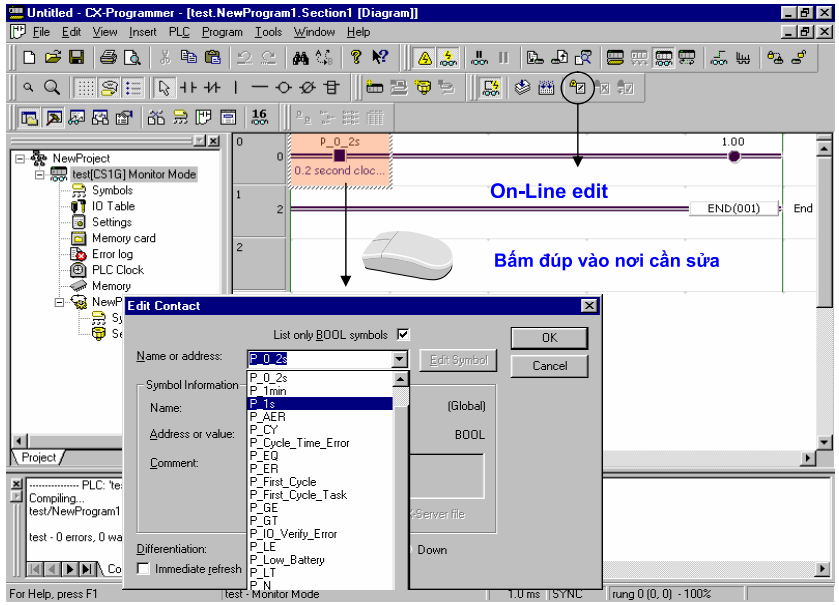


Thêm các tên (Symbol) cục bộ vào danh sách

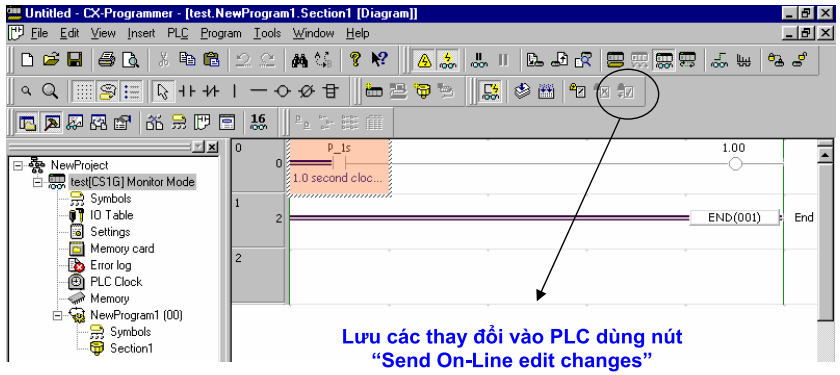


Thay đổi chương trình trực tiếp On-line

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

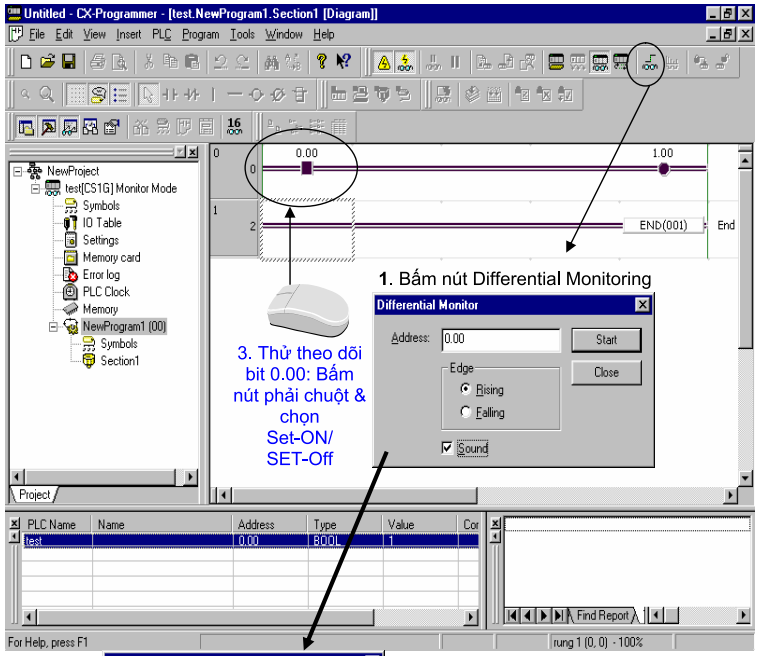


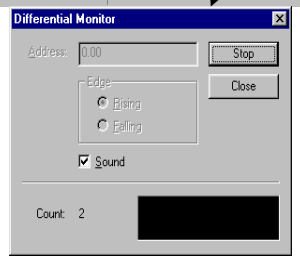
Sau khi thực hiện các thay đổi trên CX-Programmer, cần phải lưu các thay đổi này vào bộ nhớ PLC.



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

CHƯƠNG 2

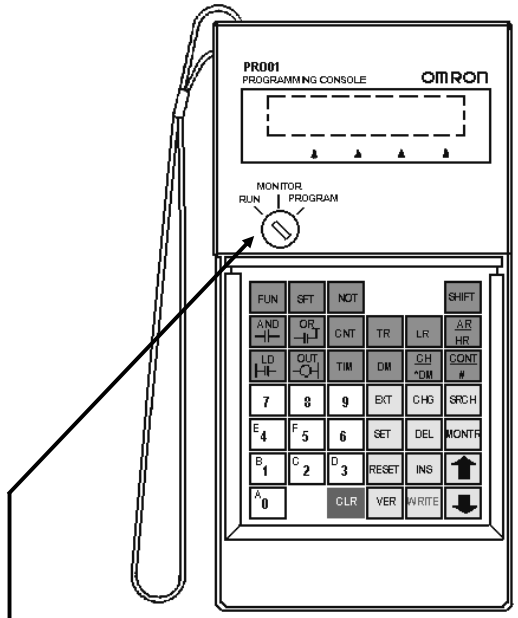
LẬP TRÌNH VỚI PROGRAMMING CONSOLE

2. Bước đầu với lập trình (Programming)

2.1 Bộ lập trình cầm tay (Programming Console)

2.1.1 Thiết lập cấu hình ban đầu (Initial Setting)

PLC có thể được đặt một trong 3 chế độ từ Programming Console bằng khoá chuyển (Key Switch) như trên hình dưới đây :



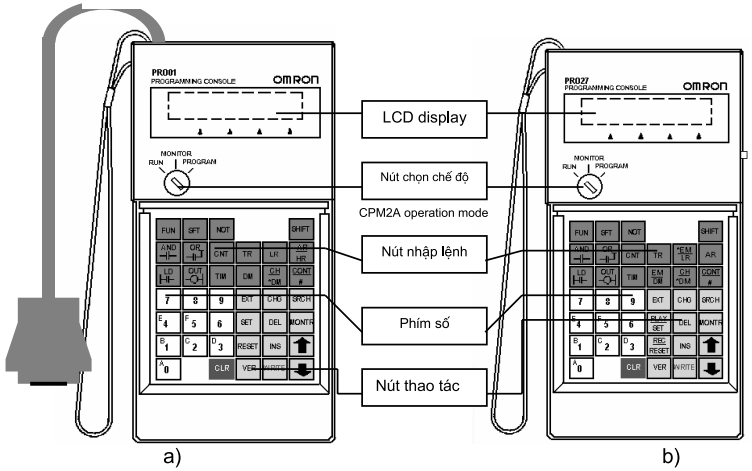
Hình 19: Programming Console loại CQM1-PRO01-E

3 chế độ làm việc của PLC

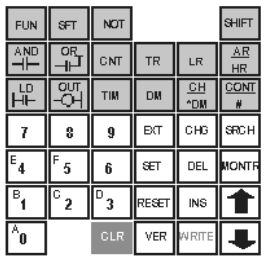
- **PROGRAM** mode : Là chế độ dùng khi viết chương trình hay thực hiện các thay đổi hoặc sửa đổi đối với chương trình hiện hành
- **MONITOR** mode : Là chế độ được dùng khi thay đổi nội dung bộ nhớ trong khi PLC đang chạy (Run).
- **RUN** mode : Là chế độ dùng để thực hiện (chạy) chương trình mà ta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

Ngoài ra, bộ CPM2A có thể được lập trình bằng Programming Console loại CQM1-PRO01-E hoặc C200H-PRO27-E. Chương trình sẽ được nhập bằng các lệnh tắt gọi là dạng Mnemonic Code. Bộ lập trình cầm tay sẽ được nối vào cổng Peripheral Port của bộ CPM2A.

CQM1-PRO01-E (có sẵn cáp nối) C200H-PRO27-E



Hình 20 : a) CQM1-PRO01-E (có cáp sẵn)
b) C200H-PRO27-E



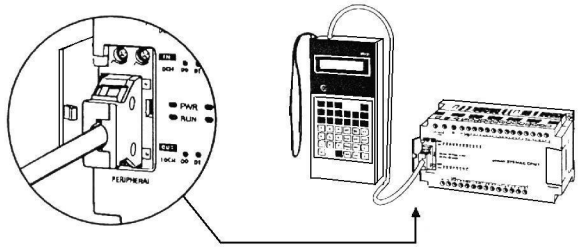
Hình 21: Các phím của bộ CQM1-PRO01-E

2.1.2 Nhập Password:

PLC có cơ chế bảo vệ bằng mật khẩu (password) để ngăn những truy cập bất hợp lệ vào chương trình bên trong PLC. PLC sẽ luôn nhắc yêu cầu người dùng nhập password khi Programming Console mới được nối với PLC và PLC đang ở chế độ hoạt động hoặc khi PLC mới được cấp điện.

<PROGRAM>
PASSWORD!

Hình 22: Màn hình LCD của Programming Console sẽ hỏi password khi muốn truy cập vào nội dung của PLC.



Hình 23: Ghép nối giữa Programming Console với PLC

Để nhập password, bấm các phím trên Programming Console theo thứ tự như sau :

CLR ⇨ MONTR ⇨ CLR ⇨ <PROGRAM> 0000

2.1.3 Các phím trên bàn phím của Programming Console

Việc lập trình cho CPM2A được lập dưới dạng mã gọi là *Mnemonic Code* với các chức năng tương đương với dạng lập trình bậc thang (*ladder diagram logic*). Trong chương trình dạng Mnemonic, mỗi lệnh được biểu diễn bằng một ký hiệu tắt (ví dụ lệnh JUMP được ký hiệu bằng JMP) và có thể kèm theo mã của lệnh (code) và các tham số lệnh như đầu vào, đầu ra, địa chỉ, ... Mỗi lệnh có một độ dài nhất định (chiếm một số word trong bộ nhớ chương trình) và khi liệt kê chương trình dưới dạng mnemonic code thường kèm theo vị trí hay địa chỉ (address) của lệnh trong chương trình ở cột bên trái ngoài cùng của lệnh.

Các phím chính trên Console :

FUN Nhiều lệnh đặc biệt gọi là Function như mov(21) có thể được nhập dùng phím này (FUN + mã lệnh)

LD Lệnh Load (LD) tạo ra 1 tiếp điểm thường mở (NO) nối vào power bus trái và là lệnh đầu tiên của nhánh. Nó cũng để bắt đầu một nhánh mới của chương trình.

AND Lệnh AND tạo ra một tiếp điểm thường mở và nối tiếp với các phần tử tiếp điểm đi trước

OR Lệnh OR tạo ra một tiếp điểm thường mở và nối song song với các phần tử đi trước tạo thành các nhánh song song.

OUT — Lệnh OUT sẽ cho ra output của nhánh chương trình (coil)

TIM — Lệnh với bộ định thời gian trễ (Timer)

CNT — Lệnh với bộ đếm (Counter)

NOT — Dùng với các lệnh LD, AND và OR để chỉ định một tiếp điểm thường đóng NC (Normally Closed)

HR — Dùng chỉ định địa chỉ bộ nhớ là loại Holding Relay

TR — Dùng chỉ định địa chỉ bộ nhớ là loại Temporary Relays

SFT — Dùng để hiển thị hoạt động với thanh ghi dịch (SHIFT Register)

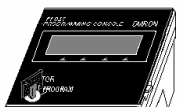
A 0 ... 9 — Dùng để nhập các số thập phân hoặc hexa khi lập trình.

SHIFT — Dùng để truy cập đến chức năng dưới đối với các phím có 2 chức năng như phím số A 0 hoặc các chức năng đặc biệt khác.

2.1.4 Xoá toàn bộ chương trình PLC

Chương trình và dữ liệu (có thuộc tính cho phép đọc/ghi) của PLC được lưu trong bộ nhớ có thể được xoá để có thể nhập vào 1 chương trình mới.

1. Chuyển PLC sang chế độ PROGRAM mode (chuyển PLC sang chế độ Program bằng cách chuyển khoá chuyển trên Programming Console về vị trí PROGRAM khi đang được nối với PLC)



(LCD)
< PROGRAM>

2. Bấm nút CLR cho đến khi trên màn hình hiện lên 00000

CLR ⇨ LCD: 00000

3. Bộ nhớ trong PLC sẽ bị xoá sau khi bấm các phím sau :

PLAY SET ⇨ NOT ⇨ REC RESET ⇨

MONTR ⇨ (LCD)
00000 MEMORY CLR
END HR CNT DM

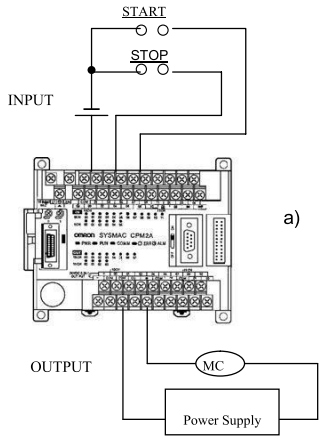
Màn hình sẽ hiện lên như trên sau khi xoá

Chú ý : Các xác lập của PLC (lưu ở dịa chỉ DM6600-DM6655) có thể bị xoá bởi lệnh này. Do vậy thật cẩn thận khi dùng.

- **Xoá một phần bộ nhớ :** Bấm các phím HR, CNT hoặc DM trước khi bấm phím MONTR để bỏ qua không xoá các vùng bộ nhớ này (HR, Counter, DM).

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

Dùng Programming Console nhập vào chương trình nhỏ sau đây với các đầu vào ra (I/O) sau :

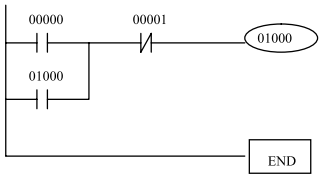


a)

Input	Thiết bị ngoài
00000	Nút bấm Start
00001	Nút bấm Stop

Output	Thiết bị ngoài
01000	Motor

Ladder Diagram



b)

Mnemonic Codes

Đ. chỉ	Lệnh	Th. số
00000	LD	00000
0001	OR	01000
0002	AND NOT	00001
0003	OUT	01000
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 01000 sẽ luôn ở trạng thái ON khi 00000 lên 1 bất kể sau đó trạng thái của đầu vào 00000 như thế nào.

2.1.6 Nhập chương trình :

1. Chuyển khoá chuyển về vị trí Program Mode. Bấm phím CLR nếu cần thiết cho đến khi 0000 được hiển thị trên màn hình :

2. CLR ⇒ LD ⇒ A 0 ⇒ WRITE

3. OR ⇒ B 1 ⇒ A 0 ⇒ A 0 ⇒ A 0 ⇒ WRITE

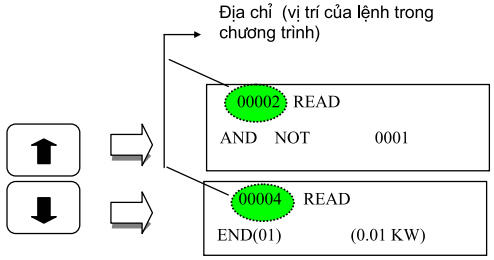
4. AND ⇒ NOT ⇒ B 1 ⇒ WRITE

5. OUT ⇒ B 1 ⇒ A 0 ⇒ A 0 ⇒ A 0 ⇒ WRITE

6. FUN ⇒ A 0 ⇒ B 1 ⇒ WRITE

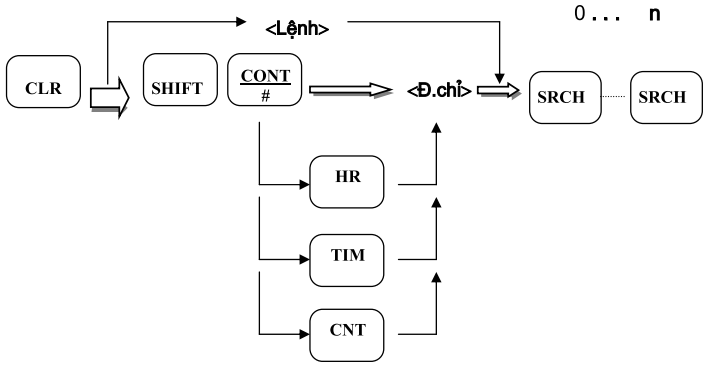
Chú ý quan trọng : Tất cả các lệnh chỉ được nạp vào bộ nhớ PLC sau khi phím **WRITE** được bấm và mỗi chương trình đều cần có một lệnh END (FUN 0).

Để chạy chương trình (Run), chuyển khoá chuyển (Key Switch) về vị trí RUN.
Để xem các bước và các lệnh của chương trình (địa chỉ chương trình), dùng phím mũi tên lên (Up Arrow) hoặc mũi tên xuống (Down Arrow)



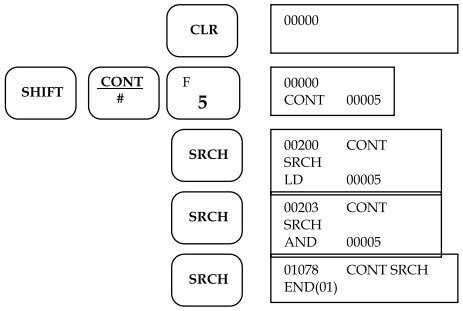
2.1.7 Tìm kiếm trong chương trình (Search)

Có thể dùng chức năng Tìm kiếm (Search) để tìm kiếm một lệnh (instruction) hoặc 1 địa chỉ bit (bit address) được dùng trong một lệnh của chương trình.
Để chỉ định một địa chỉ bit cần tìm kiếm, thực hiện như dưới đây :



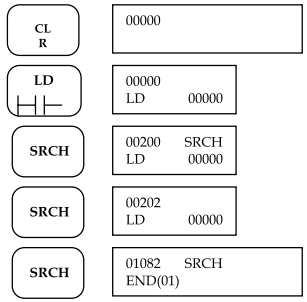
* * * n : Địa chỉ bit cần tìm kiếm

Ví dụ: Tìm kiếm bit (Bit Search) : tìm bit 00005

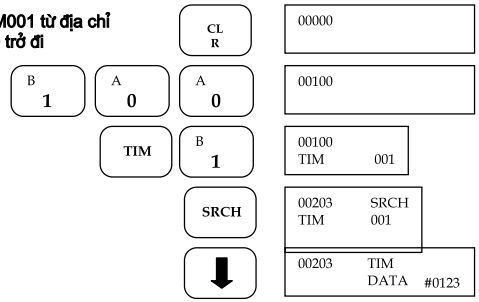


Ví dụ: Tìm kiếm lệnh (Instruction Search)

- **Tìm lệnh LD 00000**

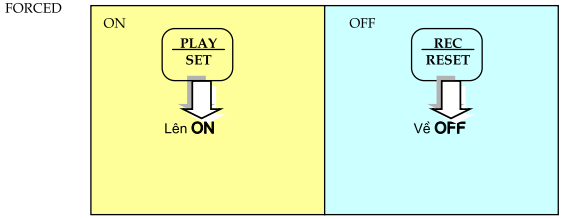


- **Tìm lệnh TIM001 từ địa chỉ 100 trở đi**



2.1.8 Bật tắt ON/OFF (SET/RESET) cưỡng bức các bit (Forced Bit Set/Reset)

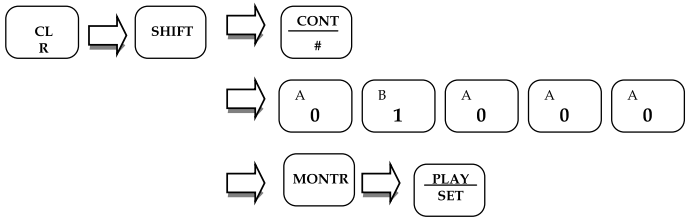
Các đầu ra số và các bit khác trong bộ nhớ có thể được bật tắt cưỡng bức lên ON hoặc về OFF không phụ thuộc vào chương trình. Việc này gọi là Forced Set / Reset.



Đặt Key Switch về chế độ MONITOR sau đó bấm các phím như hình dưới, đèn LED đầu ra 01000 sẽ:

Sáng = Đầu ra sẽ lên ON khi bấm SET

Tối = Đầu ra sẽ về OFF khi bấm RESET



2.1.9 Chèn (Insert) và xoá (Delete) các lệnh trong chương trình

✡ **Insert :**

Đặt Key Switch về vị trí Program để chèn thêm một lệnh trước một lệnh đang được hiển thị.

Việc chèn lệnh không được phép khi PLC ở chế độ RUN hay MONITOR.
Để chèn một lệnh, hiển thị lệnh mà ta muốn chèn thêm lệnh mới trước nó, sau đó nhập vào lệnh mới như bình thường và bấm các phím INS và mũi tên xuống.

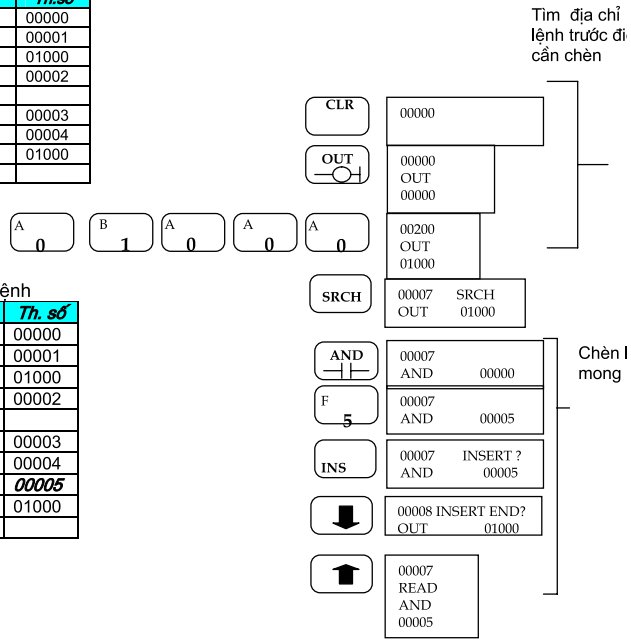
Nhập lệnh cần chèn vào ⇨ INS ⇨ ⬇

Ví dụ: Trình tự bấm các phím trên Programming Console khi chèn thêm lệnh cho 1 chương trình nhỏ dưới đây :

Chương trình dưới dạng Mnemonic Cách nhập vào bằng Programming Console

Trước khi chèn thêm lệnh

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



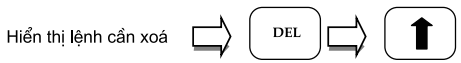
Sau khi chèn thêm lệnh

Đ. chỉ	Lệnh	Th. số
00000	LD	00000
00001	AND	00001
00002	LD	01000
00003	AND NOT	00002
00004	OR LD	
00005	AND	00003
00006	AND NOT	00004
00007	***AND***	***00005***
00008	OUT	01000
00009	END(01)	

Ở ví dụ trên, trước tiên tìm vị trí mà ta muốn chèn thêm lệnh (ở đây là sau lệnh OUT 01000) dùng phím SRCH và ra được kết quả là lệnh này ở địa chỉ 0007. Thực hiện chèn lệnh mới bằng cách nhập lệnh mới (ở đây là lệnh AND 00005) vào tại địa chỉ lệnh vừa tìm được và bấm INS + mũi tên xuống.

✡ Delete:

Để xoá lệnh đang được hiển thị, bấm phím DEL và mũi tên lên:



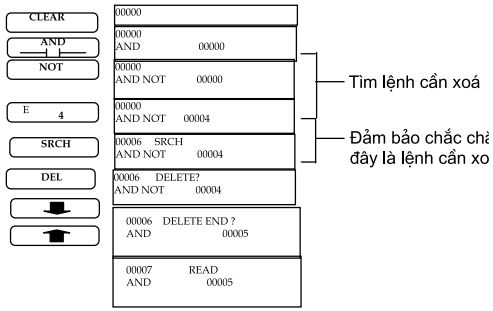
Lưu ý: Thật cẩn thận khi xoá các lệnh. Một khi lệnh đã bị xoá, lệnh sẽ không thể được phục hồi lại và ta phải nhập lại lệnh đó. Khi dùng phần mềm lập trình SYSWIN, ta có thể huỷ thao tác xoá này bằng lệnh Undo.

Ví dụ : Xoá (Delete) lệnh:

Các lệnh trước khi xoá

Đ. chỉ	Lệnh	Th. số
00000	LD	00000
00001	AND	00001
00002	LD	01000
00003	AND NOT	00002
00004	OR LD	
00005	AND	00003
00006	***AND NOT***	***00004***
00007	AND	00005
00008	OUT	01000
00009	END(01)	

Cách bấm phím



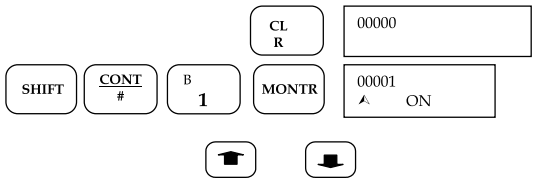
Các lệnh sau khi xoá

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

2.1.10 Theo dõi trạng thái bit trong PLC (Bit Monitor)

Trạng thái (ON/OFF) của bit bên trong bộ nhớ của PLC có thể được theo dõi khi PLC đang ở bất kỳ chế độ nào.

Ví dụ : Theo dõi trạng thái của bit 00001:

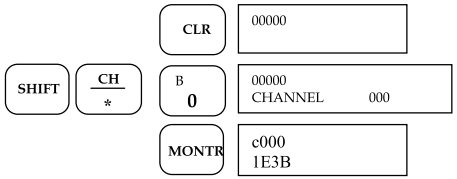


Chú ý : 1) Bấm các phím mũi tên lên hoặc xuống để xem trạng thái bit trước và sau bit hiện hành
2) Nếu PLC đang ở chế độ Program hay Monitor, ta có thể thực hiện việc Forced Set/Reset bit này.

2.1.11 Theo dõi trạng thái word (Word Monitor)

Nội dung của 1 địa chỉ dạng word có thể được theo dõi dùng chức năng Word Monitor khi PLC đang ở bất kỳ chế độ hoạt động nào.

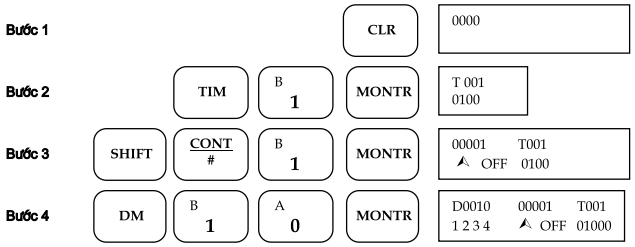
Ví dụ:



2.1.12 Theo dõi nhiều địa chỉ bộ nhớ cùng lúc (Multiple Address Monitoring)

Trạng thái/nội dung của tối đa là 6 bit hay word bộ nhớ có thể được theo dõi cùng lúc, tuy nhiên tại một thời điểm chỉ có 3 địa chỉ được hiển thị trên màn hình mà thôi.

Ví dụ: Cách theo dõi nhiều địa chỉ bộ nhớ cùng lúc: TIM001, 00001, DM0010

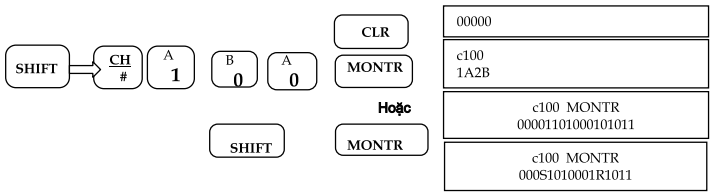


Chú ý :
- Nếu cần theo dõi cùng lúc nhiều hơn 3 bit hay word, các bit hay word hiện chưa được hiển thị trên màn hình có thể được hiển thị bằng cách bấm phím MONTR.
- Nếu theo dõi nhiều hơn 6 bit hay word, bit hay word đầu tiên sẽ không được theo dõi nữa.
- Bấm phím CLR để dừng việc theo dõi bit hay word bên trái nhất và loại nó khỏi màn hình theo dõi.
- Bấm tổ hợp phím SHIFT + CLR để dừng toàn bộ việc theo dõi.

2.1.13 Theo dõi trạng thái từng bit trong word (Binary Monitor)

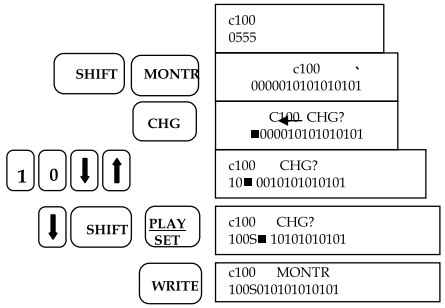
Mặc dù các địa chỉ bộ nhớ dạng word thường được xử lý như là giá trị của một tập các bit thống nhất, trạng thái ON/OFF của từng bit trong 16 bit của một word bất kỳ có thể được theo dõi khi PLC đang ở bất kỳ chế độ hoạt động nào.

Ví dụ : Cách theo dõi trạng thái từng bit trong word.



Trạng thái của bit đang bị forced-set được biểu thị bằng chữ "S" và trạng thái của bit bị forced-reset được biểu với chữ "R".

Chú ý :
- Bấm phím mũi tên lên hoặc xuống để theo dõi trạng thái của bit trước hoặc tiếp theo trong word.
- Trạng thái của bit đang được hiển thị có thể được thay đổi bằng cách sử dụng chức năng Binary Data Modification như dưới đây:

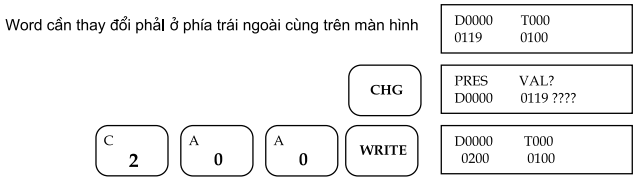


Một con nháy sẽ được hiển thị trên bit 15 để biểu thị bit có thể được thay đổi.

Chú ý:
- Dùng phím mũi tên lên hoặc xuống để dịch con nháy sang trái hoặc sang phải.
- Dùng các phím số 1 và 0 để thay đổi trạng thái của bit về ON hay OFF. Con nháy sẽ dịch sang bit kế tiếp bên phải sau khi các phím này được bấm.
- Dùng tổ hợp phím SHIFT + SET hoặc SHIFT + RESET để force-set hoặc force-reset trạng thái một bit. Bấm phím NOT sẽ xoá trạng thái Force-set hay Force-reset này.

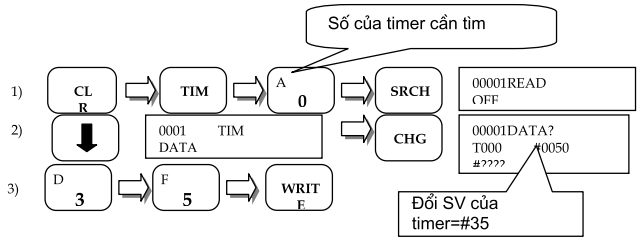
2.1.14 Thay đổi dữ liệu HEX/BCD (HEX/BCD Data Modification)

Giá trị HEX/BCD của một word đang được theo dõi có thể được thay đổi dùng tính năng này. Chức năng này chỉ có hiệu lực ở chế độ Monitor hoặc Program.



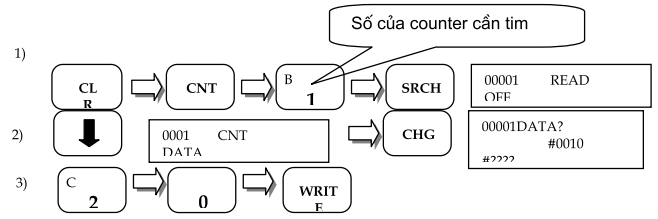
2.1.15 Thay đổi giá trị Timer

Chuyển Key Switch trên Programming Console về vị trí Monitor khi muốn thực hiện hoạt động này :



2.1.16 Thay đổi giá trị Counter :

Counter (CNT) là một bộ đếm giảm có giá trị đặt trước (preset decremental counter). Bộ đếm sẽ giảm đi 1 đơn vị mỗi khi tín hiệu đầu vào chuyển từ trạng thái OFF lên ON. Bộ đếm có các thông số cần có khi lập trình là: đầu vào xung đếm (count input), đầu vào xoá (reset input), số của bộ đếm và giá trị đặt (Set Value - SV).
Chuyển Key Switch trên Programming Console về vị trí Monitor khi muốn thực hiện hoạt động này :



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.

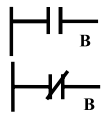
2.2.1 Lệnh điều kiện khởi đầu : Load (LD) và Load Not (LD NOT)

Lệnh LOAD hay LOAD NOT 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)** →

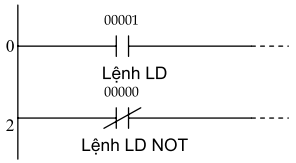
LOAD NOT-**LD NOT (Normally Closed)** →



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

Các địa chỉ có thể truy cập ở dạng bit

Ví dụ :



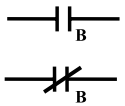
Địa chỉ	Lệnh	Th. số
00000	LD	00000
00001	Lệnh khác...	.....
00002	LD NOT	00000
00003	Lệnh khác	.....

2.2.2 Lệnh 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**

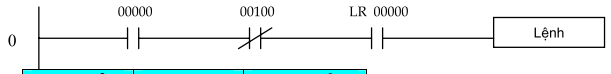
AND NOT-**AND NOT**



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

Các địa chỉ có thể truy cập ở dạng bit

Ví dụ: AND, AND NOT



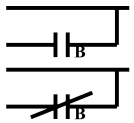
Địa chỉ	Lệnh	Th. Số
00000	LD	00000
00001	AND NOT	00100
00002	AND	LR 00000
00003	Lệnh ..	

2.2.3 Lệnh 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**

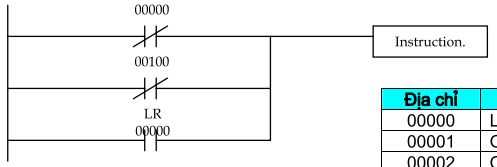
OR NOT-**OR NOT**



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

Các địa chỉ có thể truy cập ở dạng bit

Ví dụ : OR, OR NOT



Địa chỉ	Lệnh	Tham số
00000	LD NOT	00000
00001	OR NOT	00100
00002	OR	LR 00000
00003	Instruction	

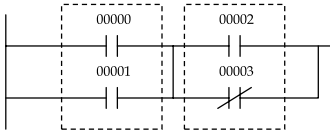
2.2.4 Lệnh AND LD và OR LD

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

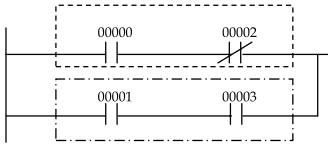
Cần chú ý thứ tự nhập lệnh này: các khối logic cần nối với nhau được nhập riêng rẽ trước, sau đó mới nhập lệnh OR LD hoặc AND LD.
Lệnh này không cần tham số.

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

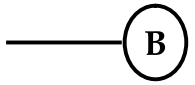


Địa chỉ	Lệnh	Th. số
00000	LD	00000
00001	AND NOT	00001
00002	LD	00002
00003	AND	00003
00004	OR LD	
00005	Lệnh ...	.....

2.2.5 Lệnh 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

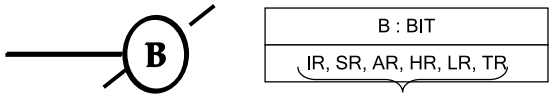
Các địa chỉ có thể truy cập ở dạng bit

Ví dụ: Lệnh OUT

Đ. chỉ	Lệnh	Th. số
00000	LD	00000
00001	OUT	01000

Tiếp điểm 00000 là điều kiện thực thi của cuộn dây 01000.

Ký hiệu: OUTPUT NOT-**OUT NOT**



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	01000

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

Ngoài các lệnh điều kiện và đầu ra đơn giản trên, trong PLC loại CPM2A còn có các lệnh với các chức năng phức tạp khác. Mỗi lệnh này đều có một mã lệnh (code) riêng. Khi lập trình với Mnemonic Code dùng Programming Console, ta phải nhập lệnh dưới dạng sau :

"FUN xx" trong đó xx : Mã của lệnh

Phím FUN và các phím số trên Programming Console dùng để nhập các lệnh đặc biệt này. Dưới đây là mã của một số lệnh trong PLC loại CPM2A :

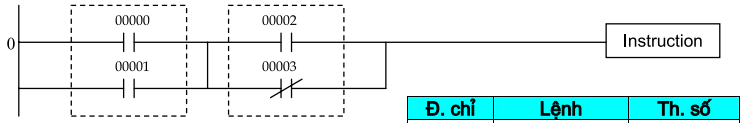
FUN 01	là lệnh	END (End Instruction)
FUN 02	,,	IL (Interlock)
FUN 03	,,	ILC (Interlock Clear)
FUN 04	,,	JMP (Jump End)
FUN 05	,,	JME (Jump End)
FUN 10	,,	SFT (Shift Register)
FUN 11	,,	KEEP (Latching Relay)
FUN 12	,,	CNTR (Reversible Counter)
FUN 13	,,	DIFU (Differentation - Up)
FUN 14	,,	DIFD (Differentation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

2.3.1 Lệnh END (FUN 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:

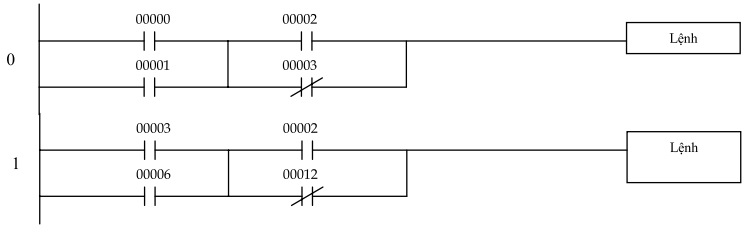


Đ. chỉ	Lệnh	Th. số
00000	LD	00000
00001	OR	00001
00002	LD	00002
00003	OR NOT	00003
00004	AND LD	---

Trên màn hình LCD:

Rung (network)

Các phần tử khi kết nối với nhau sẽ tạo thành 1 thang (gọi là rung hay network) trong sơ đồ bậc thang.
Ở ví dụ dưới, ta có 2 thang độc lập đánh số theo thứ tự là 0 và 1.
Khi sửa hay thêm chương trình, ta cần chú ý đặt các phần tử vào đúng thang.



2.3.2 Lệnh IL (FUN 02) và ILC (FUN 03)

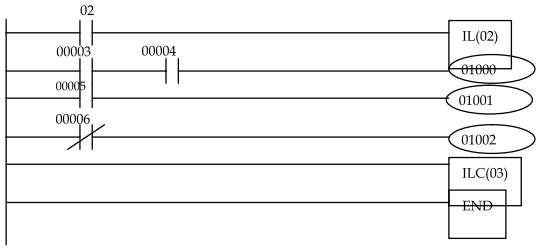
Lệnh IL (Interlock) và ILC (Interlock Clear) luôn được dùng đi kèm với nhau. Khi một lệnh IL được đặt trước một đoạn chương trình, điều kiện thực hiện của IL sẽ điều khiển điều kiện thực hiện của toàn bộ các lệnh bắt đầu từ sau lệnh IL cho đến lệnh ILC đầu tiên sau lệnh IL này. Khi điều kiện thực hiện 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:

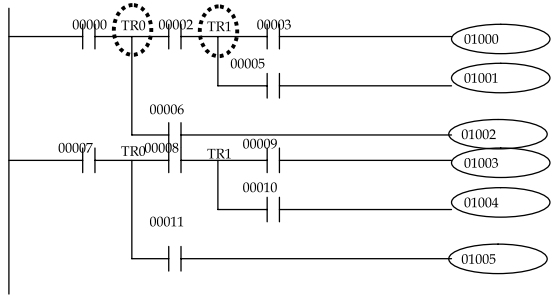
Đ. chỉ	Lệnh	Th. số
00000	LD	00002
00001	IL (02)	-
00002	LD	00003
00003	AND	00004
00004	OUT	01000
00005	LD	00005
00006	OUT	01001
00007	LD NOT	00006
00008	OUT	01002
00009	ILC(03)	-
00010	END(01)	-

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 không bị reset khi điều kiện thực thi của IL là OFF.

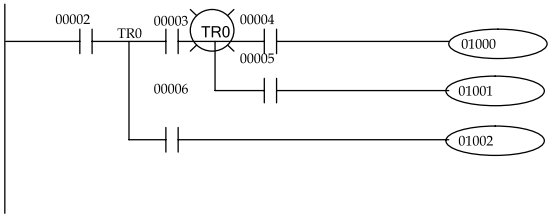


2.3.3 Bit phân nhánh - TR (Temporary Relay)

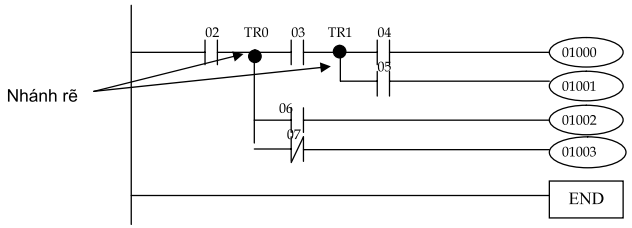
Trong các nhánh chương trình, các bit phân nhánh (7 bit từ TR0-TR7) được dùng để lưu điều kiện thực hiện tại điểm phân nhánh, giúp cho việc thực hiện chương trình tại nhánh chương trình được đúng đắn.



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 khi nhập vào bằng bộ lập trình cầm tay. 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:

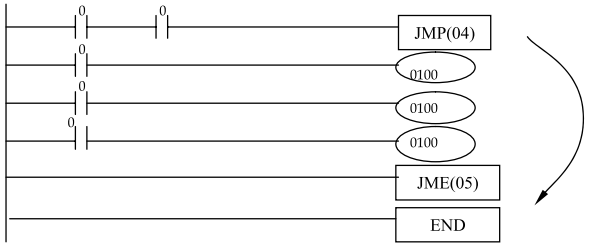
Địa chỉ	Lệnh	Th. số
0000	LD	00002
0001	OUT	TR 0
0002	AND	00003
0003	OUT	TR1
0004	AND	00004
0005	OUT	01000
0006	LD	TR1
0007	AND	00005
0008	OUT	01001
0009	LD	TR0
0010	AND	00006
0011	OUT	01002
0012	LD	TR0
0013	AND-NOT	00007
0014	OUT	01003
0015	END (01)	-

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

2.3.4 Lệnh JMP (FUN 04) và JME (FUN 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 dụ : Lệnh JUMP



Chương trình dạng mnemonic :

Đ. chỉ	Lệnh	Th. số
0000	LD	00002
0001	AND	00003
0002	JMP(04)	
0003	LD	00004
0004	OUT	01000
0005	LD	00005
0006	OUT	01001
0007	LD	00006
0008	OUT	01002
0009	JME(05)	
0010	END(01)	

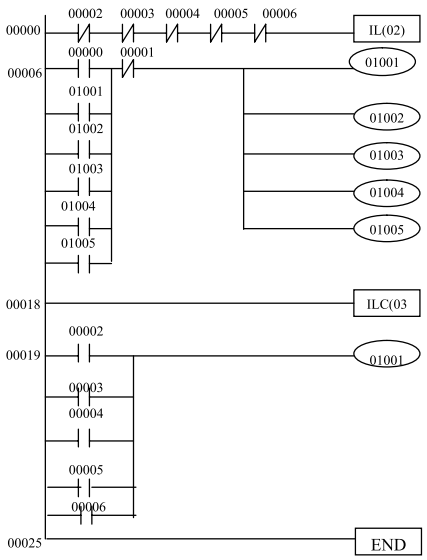
2.4 Ví dụ ứng dụng: Dừng động cơ khi có quá tải

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	01000	Lamp Overload
00001	PB Stop	01001	Motor 1
00002	Overload M1	01002	Motor 2
00003	Overload M2	01003	Motor 3
00004	Overload M3	01004	Motor 4
00005	Overload M4	01005	Motor 5
00006	Overload M5		

Chương trình dang sơ đồ bâc thang



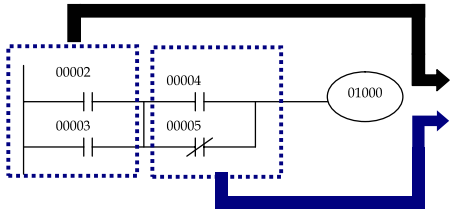
Chương trình dang Mnemonic

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		01001
00008	OR		01002
00009	OR		01003
00010	OR		01004
00011	OR		01005
00012	AND	NOT	00001
00013	OUT		01001
00014	OUT		01002
00015	OUT		01003
00016	OUT		01004
00017	OUT		01005
00018	ILC	(03)	
00019	LD		00002
00020	OR		00003
00021	OR		00004
00022	OR		00005
00023	OR		00006
00024	OUT		01001
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:*

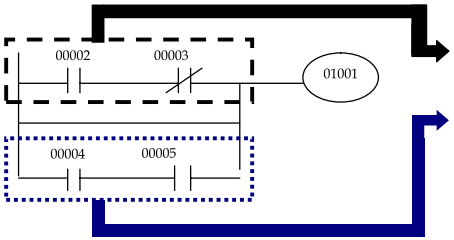
Ví du : Cách nhập các lệnh AND LD và OR LD với bộ lập trình cầm tay

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



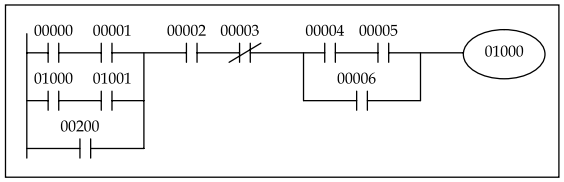
Đ. chỉ	Lệnh	Th. số
00000	LD	00002
00001	OR	00003
00002	LD	00004
00003	OR NOT	00005
00004	AND LD	- - -
00005	OUT	01000

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



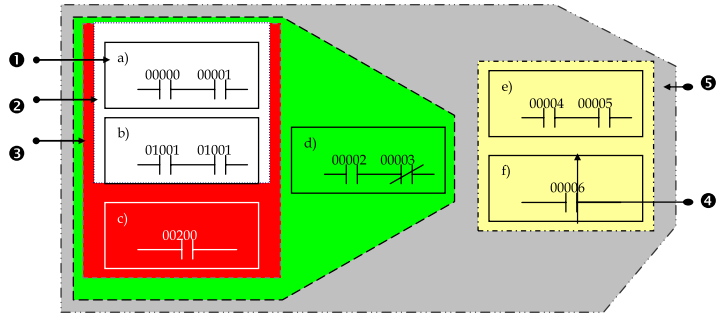
Đ. chỉ	Lệnh	Th. số
00000	LD	00002
00001	AND NOT	00003
00002	LD	00004
00003	AND	00005
0004	OR LD	- - -
00005	OUT	01001

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

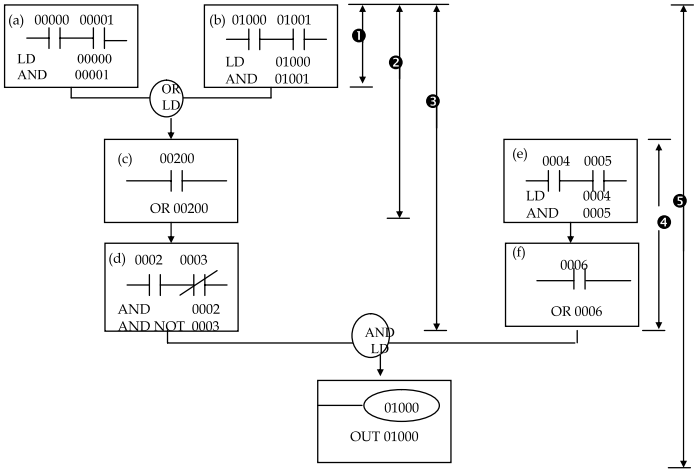


Để có thể nhập được chương trình này cũng như các khối chương trình phức tạp khác vào bằng bộ lập trình cầm tay, cần thực hiện các bước sau :

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:

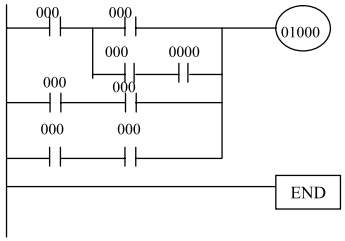


Chú ý : Các khối logic cơ bản là các khối với các phần tử có thể được nối với nhau bằng các lệnh LD, LD NOT, AND, AND NOT, OR, OR NOT, ..

Bài ôn tập :

Cho một chương trình dưới dạng Ladder Diagram dưới đây. Hãy nhập vào bằng bộ lập trình cầm tay dưới dạng Mnemonic Code :

Ladder Diagram

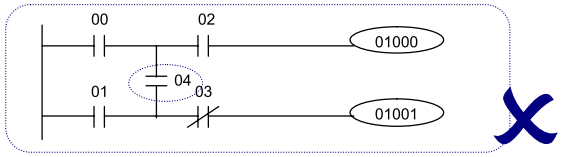


Mnemonic Codes

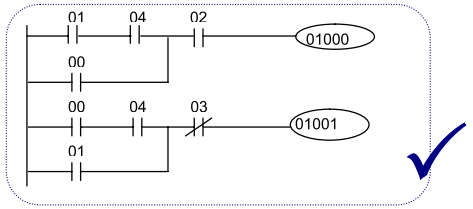
Đ. chỉ	*Lệnh*	*Th.số*

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

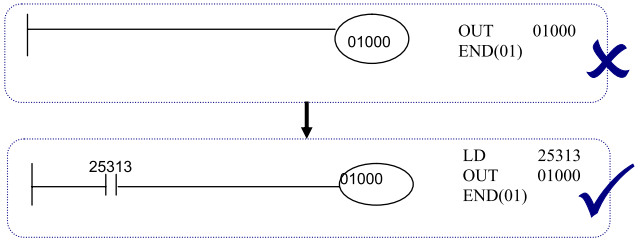
1. Hai thang khác nhau không được phép nối bằng một tiếp điểm thẳng đứng :



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 :

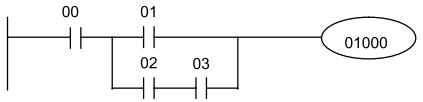


2. Nếu một lệnh OUTPUT hoặc một FUN luôn luôn cần điều kiện thực hiện là ON, lệnh này không được nối trực tiếp với thanh power bus bên trái. Thay vào đó, phải nối qua một tiếp điểm dùng cờ "ALWAYS ON" (có địa chỉ là 25313)



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.



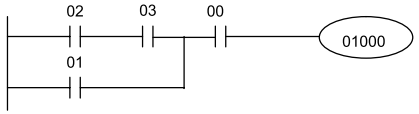
Hình A :

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	01000

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

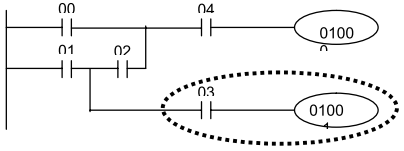


Các lệnh dạng Mnemonic cho sơ đồ hình B

Đ. chỉ	*Lệnh*	*Th. số*
0000	LD	02
0001	AND	03
0002	OR	01
0003	AND	00
0004	OUT	01000

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.

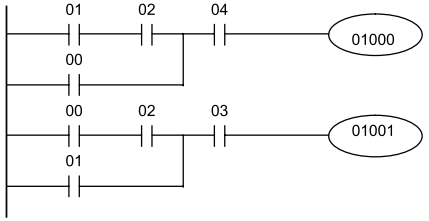


Khi các bit 01, 02, 04 là ON sẽ bật 1000 lên ON

Hình A

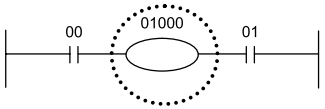
Ở hình A, ta muốn khi các bit 00, 02 và 03 là ON hoặc khi 01 và 03 là ON, bit 01001 sẽ được bật lên ON. Tuy nhiên đó là cách biểu diễn không thích hợp với việc nhập bằng Console.

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

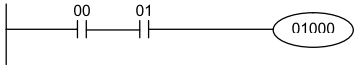


Hình B

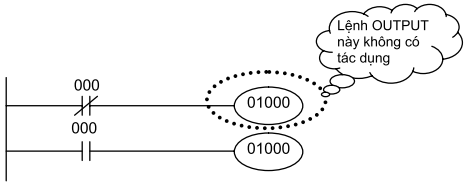
5 Lệnh OUT hoặc OUT NOT (nếu có) phải là lệnh cuối cùng trên thang và phải được nối vào power bus bên phải.



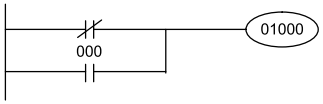
Đoạn chương trình trên không đúng vì lệnh OUT 01000 không được nối trực tiếp vào power bus mà qua một tiếp điểm và sẽ được sửa lại như sau :



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

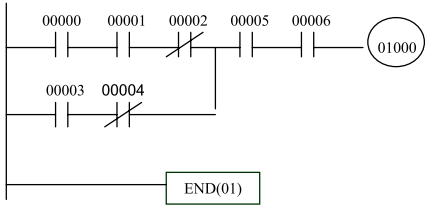


Do đó, nếu 2 bit 00000 và 00011 đều dùng để điều khiển lệnh OUTPUT với bit 01000 thì đoạn chương trình trên sẽ được sửa lại như sau :



Bài ôn tập về Lập trình cơ bản trên CPM2A dùng Programming Console

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

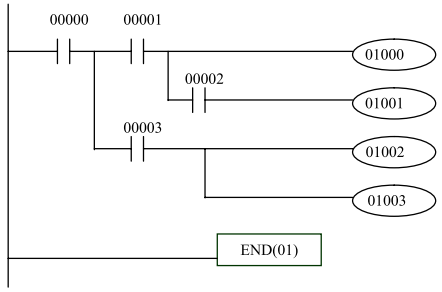


Đ. chỉ	Lệnh	Th. số
00000		
00001		
00002		
00003		
00004		
00005		
00006		
00007		
00008		
00009		

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	01000
00012	END (01)	

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



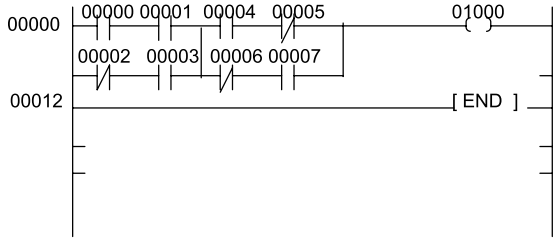
Đ. chỉ	Lệnh	Th. số
00000		
00001		
00002		
00003		
00004		
00005		
00006		
00007		
00008		
00009		
00010		

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         01000
00009  END(01)
```

2)



3)

```
00000  LD          00000
00001  OUT     TR      0
00002  AND         00001
00003  OUT         01000
00004  AND         00002
00005  OUT         01001
00006  LD      TR      0
00007  AND         00003
00008  OUT         01002
00009  OUT         01003
00010  END(01)
```

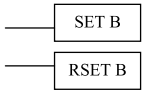
Chú ý : Nhánh rẽ với lệnh AND 00002 và OUT 00001 không cần thêm bit TR vì giữa điểm rẽ nhánh và lệnh OUT 01000 không có tiếp điểm nào.

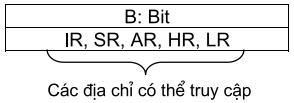
2.6 Các lệnh đặc biệt thông dụng khác:

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.

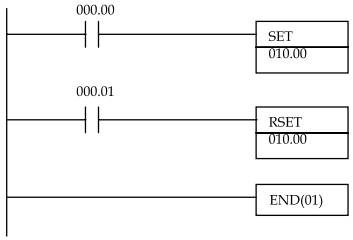
Ký hiệu





Ví dụ: Bit 01000 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 01000 sẽ vẫn giữ (Hold) ở trạng thái ON cho dù bit 00000 là ON hay OFF. Bit 01000 sẽ chỉ bị xoá bởi lệnh Reset khi bit 00001 là ON.

Sơ đồ bậc thang



Các lệnh dạng Mnemonic

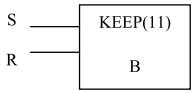
```
00000  LD      000.00
00001  SET     010.00
00002  LD      000.01
00003  RSET    010.00
00004  END
```

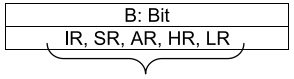
Chú ý : Trạng thái của bit được SET hay RSET sẽ không thay đổi khi nằm trong khối INTERLOCK hay JUMP.

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.

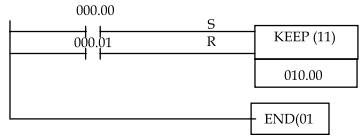
Ký hiệu





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

Ví dụ : Bit 01000 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 01000 chỉ bị reset về OFF khi bit 00001 là ON (đầu vào RESET sẽ tác động)

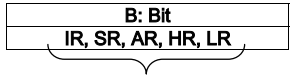



```
00000  LD      000.00
00001  LD      000.01
00002  KEEP    010.00
00003  END
```

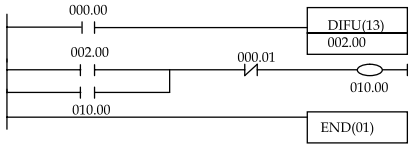
2.6.3 DIFFERENTIATE UP và DOWN - DIFU(13) & DIFD(14)

- **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

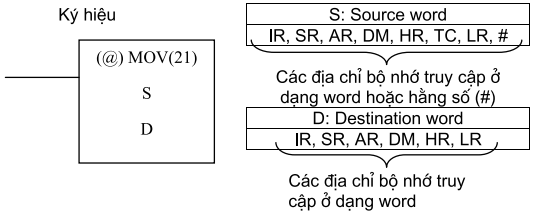


Ví dụ: Khi bit 000.00 chuyển từ OFF ở chu kỳ quét trước lên ON ở chu kỳ quét hiện hành, bit 002.00 sẽ được bật lên ON trong vòng một chu kỳ. Ở chu kỳ quét sau, bit 002.00 lại được quay trở về OFF.



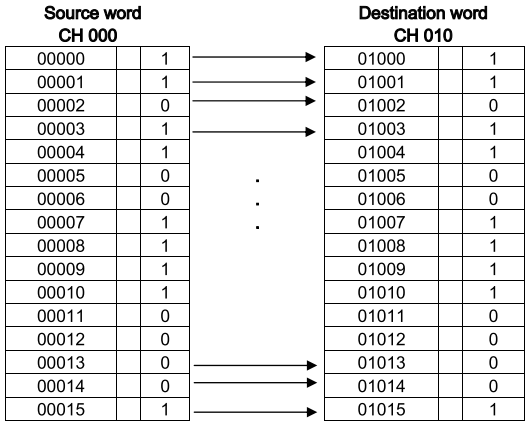

```
00000  LD      000.00
00001  DIFU    002.00
00002  LD      002.00
00003  OR      010.00
00004  AND NOT 000.01
00005  OUT     010.00
00006  END
```

2.6.4 Lệnh copy dữ liệu MOVE - MOV(21)



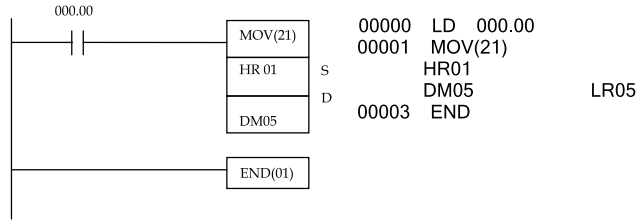
S = Là địa chỉ của word nguồn (Source word) hoặc một hằng số
(# là ký hiệu của một hằng số, ví dụ #155,...được nhập vào ngay khi lập trình)
D = Là địa chỉ của word đích (Destination word)

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:

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)

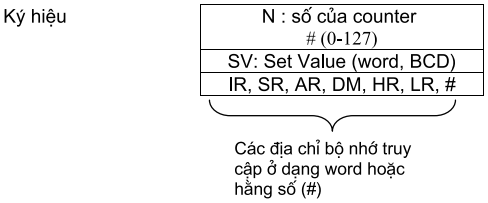
Chú ý : Mỗi bộ counter và timer có một số duy nhất từ 0 đến 127 và không được phép dùng trùng lặp trong lệnh đếm/timer khác của chương trình.

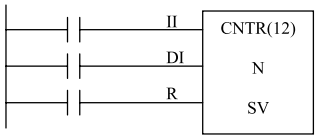
Số của bộ đếm và timer có 2 cách dùng như sau :

- 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).

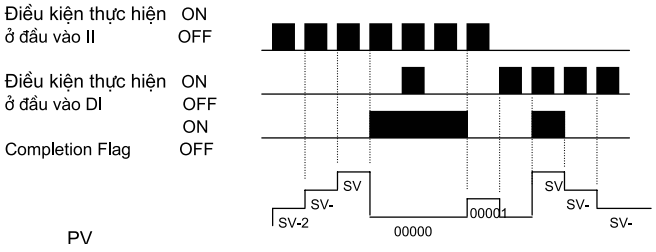
CNTR là một bộ đếm có thể đếm theo hai chiều tăng - giảm:

- 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.



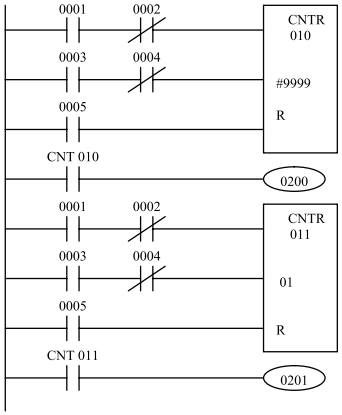


- II : Đầu vào đếm tăng
- DI : Đầu vào đếm giảm
- R : đầu vào reset giá trị PV
- SV : Giá trị đặt trước



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

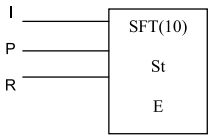
Ladder Diagram



Mnemonic Code

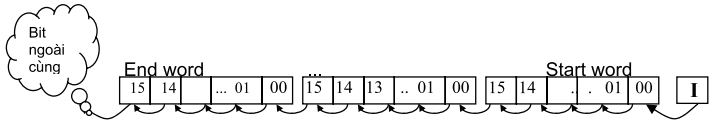
Đ. chỉ	Lệnh	Th. số
0200	LD	0001
0201	AND-NOT	0002
0202	LD	0003
0203	AND-NOT	0004
0204	LD	0005
0205	CNTR(12)	010
		# 9999
0206	LD	CNT 010
0207	OUT	0200
0208	LD	0001
0209	AND-NOT	0002
0210	LD	0003
0211	AND-NOT	0004
0212	LD	0005
0213	CNTR(12)	011
		01
0214	LD	CNT 011
0215	OUT	0201

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



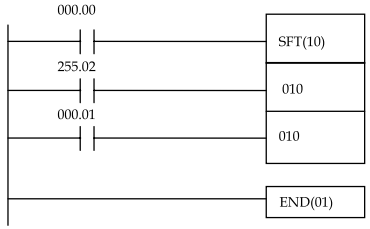
St = Word đầu tiên của thanh ghi dịch
E = Word cuối của thanh ghi dịch
I = Bit dịch (Input bit)
P = Bit xung nhịp ((Shifting) Pulse Input)
R = Đầu vào xoá (Reset Input)

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:

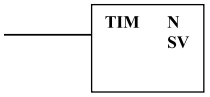
Ở ví dụ dưới đây, ta có 1 thanh ghi dịch dài 1 word (St= 010, E =010) tại địa chỉ 010. Lệnh SFT(10) sẽ dịch các bit của của thanh ghi dịch sang bên trái một vị trí bit và bit 000.00 được dịch vào bit ngoài cùng bên phải (tức bit 010.00) của thanh ghi này mỗi khi bit 255.02 chuyển từ OFF lên ON. Bit 25502 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, #

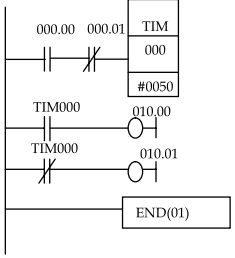
Các địa chỉ bộ nhớ truy cập ở dạng word hoặc hằng số (#)

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

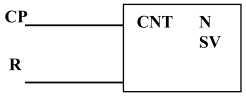


Mnemonic Code

```
00000   LD          000. 00
00001   AND NOT     000. 01
00002   TIM         000
                    #0050
00003   LD          TIM000
00004   OUT         010. 00
00005   LD  NOT     TIM000
00006   OUT         010. 01
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 (#)

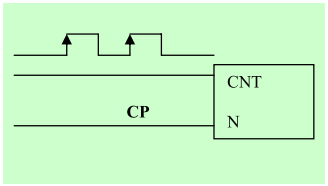
SV: Set Value (Word, BCD)
IR, SR, AR, DM, HR, LR, #

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

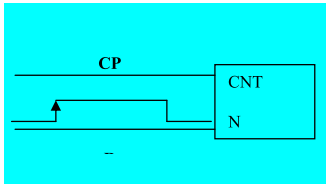
- N : Số của bộ đếm (từ 000 đến 127)
- SV : Giá trị đặt (từ 0 đến 9999) và phải ở dạng BCD
- CP : Đầu vào xung đếm
- R : Đầu vào reset

Ví dụ:

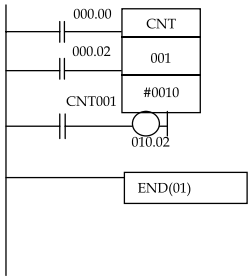
Bộ đếm đang đếm



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 010.02 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         010.02
00005   END
```

2.6.9 Ví dụ về ứng dụng COUNTER và TIMER

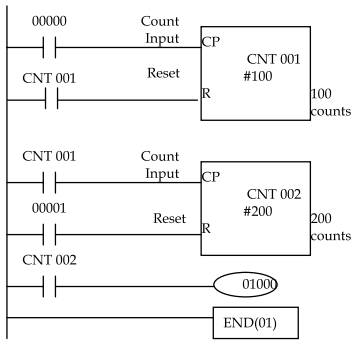
Ví dụ 1 Mở rộng khả năng đếm của counter

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	01000
00010	END(01)	

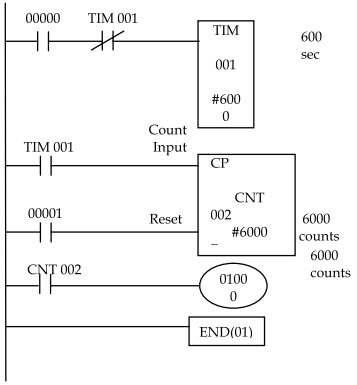
Ví dụ 2 Kéo dài thời gian trễ của timer lên 1.000 giờ

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	01000

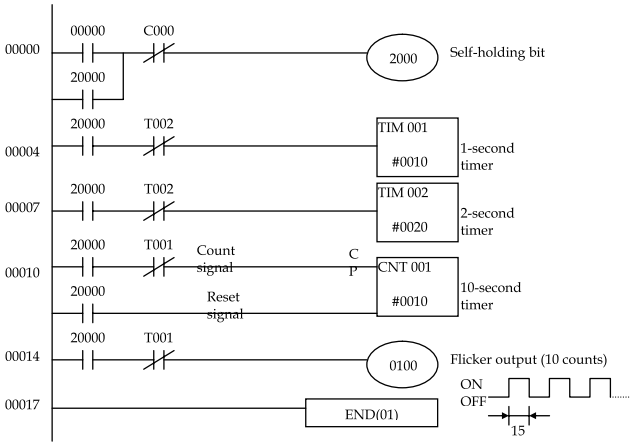
Ladder



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	01000
00008	END (01)	

Ví dụ 3 Chương trình này sẽ làm nhấp nháy (flicker) đầu ra 010.00 (bật 1 giây, tắt 1 giây) ON/OFF 10 lần sau khi bit 000.00 lên ON.

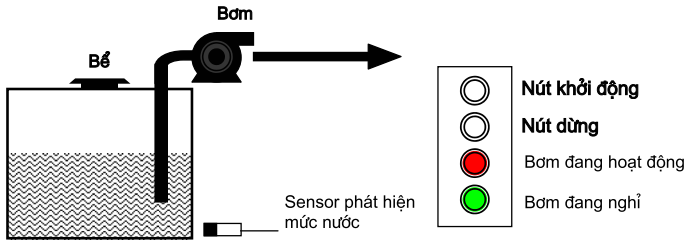


Đ. chỉ	Lệnh	Th. số		Chú thích
00000	LD		00000	(1) Self-holding bit
00001	OR		20000	
00002	AND NOT	C*	000	
00003	OUT		20000	
00004	LD		20000	(2) 1-Second timer
00005	AND NOT	T**	002	
00006	TIM		001	
		#	0010	
00007	LD		20000	(3) 2-Second timer
00008	AND NOT	T	002	
000009	TIM		002	
		#	0020	
00010	LD		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		01000	
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.

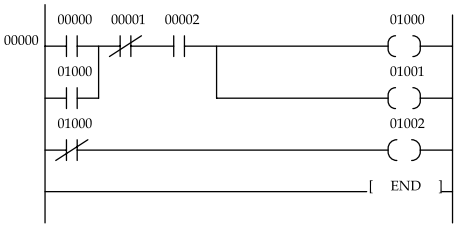


Chương 2

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	01000	Đầu ra điều khiển bơm
	01001	Đèn báo bơm đang chạy
	01002	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	01000
00005	OUT	01001
00006	LD NOT	01000
00007	OUT	01002
00008	END (01)	