

SỞ LAO ĐỘNG – THƯƠNG BINH VÀ XÃ HỘI HÀ NỘI
TRƯỜNG TRUNG CẤP CÔNG NGHỆ VÀ DU LỊCH HÀ NỘI



GIÁO TRÌNH
MÔ ĐUN: PCL CĂN BẢN
NGHỀ: CÔNG NGHỆ KỸ THUẬT ĐIỆN – ĐIỆN TỬ
TRÌNH ĐỘ: TRUNG CẤP

*(Ban hành kèm theo Quyết định số: 32/QĐ-CNDL ngày 28 tháng 02 năm 2023
của Hiệu trưởng Trường Trung cấp Công nghệ và Du lịch Hà Nội)*

Hà Nội, năm 2023

GIỚI THIỆU

PLC (Programmable Logic Control) là các thiết bị điều khiển logic lập trình được. Do có nhiều ưu thế cả về kỹ thuật lẫn kinh tế nên PLC hiện nay đang được dùng rất phổ biến trong việc điều khiển các hệ thống tự động trong các nhà máy, xí nghiệp và các ứng dụng khác của đời sống. Đó là lý do PLC được đưa vào giảng dạy chính thức trong chương trình của hệ Cao đẳng và trung cấp nghề, chuyên ngành Điện công nghiệp, Kỹ thuật máy lạnh và Điều hòa không khí. Hiện nay, tài liệu về PLC không hiếm, tuy nhiên các tài liệu này được biên soạn với nhiều mục đích khác nhau do đó cũng rất đa dạng. Có những tài liệu thuần về lý thuyết, cũng có những tài liệu đi sâu vào việc hướng dẫn người học cách lập trình và mô phỏng một số bài tập ứng dụng, bên cạnh đó còn có các tài liệu có nội dung rất bao quát, trình bày từ những kiến thức hết sức căn bản đến các nội dung nâng cao vượt ngoài khả năng hiểu biết của đối tượng sinh viên cao đẳng và học sinh trung cấp. Các tài liệu này đa phần đều không bám sát chương trình chi tiết đang áp dụng tại trường nên trong quá trình giảng dạy, giáo viên thường phải sưu tầm nhiều nguồn tài liệu khác nhau và ở mỗi tài liệu, giáo viên trích ra một vài nội dung để giảng dạy. Thực trạng này làm cho cấu trúc chương trình môn học không mang tính khoa học và cũng tạo nên sự không thống nhất giữa các giáo viên cùng dạy một môn học. Sau vài năm giảng dạy và nghiên cứu lĩnh vực PLC, tôi đã biên soạn nên tập “Giáo trình PLC cơ bản” với chủ đích như sau:

- Hình thành một tài liệu mang tính thống nhất về nội dung và hình thức cho môn học PLC.

- Tài liệu này có nội dung bám sát chương trình chi tiết đang áp dụng tại trường.

- Các kiến thức trong tài liệu đảm bảo tính chính xác, khoa học và được diễn đạt bằng ngôn từ dễ hiểu.

- Nội dung tài liệu rõ ràng, xúc tích, hình ảnh minh họa dễ hiểu, giúp người học có thể tự học tập, nghiên cứu thêm.

Giáo trình này gồm các nội dung chính sau:

Bài 1: Đại cương về điều khiển lập trình. Bài

2: Cài đặt và sử dụng phần mềm S7-300 Bài

3: Các phép toán nhị phân của PLC.

Bài 4: Các phép toán số của PLC

Bài 5: Kết nối hệ thống điều khiển PLC

Đây là giáo trình được biên soạn dựa trên các tài liệu có sẵn và kiến thức của bản thân. Trong quá trình biên soạn khó tránh khỏi những thiếu sót, rất mong được sự góp ý chân thành của đồng nghiệp và các nhà chuyên môn nhằm giúp cho giáo trình được cập nhật, chỉnh sửa để ngày càng hoàn thiện hơn trong thời gian tới.

Chân thành cảm ơn./.

Hà Nội, ngày tháng năm 2023

MỤC LỤC

GIỚI THIỆU	1
MỤC LỤC.....	2
CHƯƠNG TRÌNH MÔ ĐUN PLC CƠ BẢN.....	6
Bài 1: ĐẠI CƯƠNG VỀ ĐIỀU KHIỂN LẬP TRÌNH.....	7
I. Cấu trúc cơ bản của hệ thống điều khiển.....	7
1. Tổng quan về các hệ thống điều khiển.	7
2. Tín hiệu vào:	8
3. Bộ điều khiển.	8
II. Giới thiệu về PLC.....	10
1. Sơ lược về lịch sử phát triển:.....	10
2. Cấu trúc cơ bản và hoạt động của một PLC.....	11
III. PLC S7-300.....	20
1. Các tính năng của PLC S7-300	20
2. Cấu trúc phần cứng PLC S7-300:.....	20
3. Cấu trúc bộ nhớ PLC S7-300:.....	24
Bài 2: CÀI ĐẶT VÀ SỬ DỤNG PHẦN MỀM S7-300	33
I. Cài đặt phần mềm Step7	33
A. PHẦN LÝ THUYẾT:	33
1. Giới thiệu phần mềm Step7	33
2. Các bước cài đặt phần mềm Step7	34
B. PHẦN THỰC HÀNH:.....	37
II. Sử dụng phần mềm Step7:.....	37
A. PHẦN LÝ THUYẾT:	37
1. Khởi động chương trình tạo project:	37
1.1. Cách 1: Lưu Project mặc định trong ổ C:/Program Files/... ..	37
1.2. Cách 2: Lưu Project vào các ổ đĩa khác:	40
2. Thiết lập phần cứng:	43
3. Viết chương trình điều khiển:	45
4. Mô phỏng chương trình	47
5. Download chương trình lên PLC.....	48

B. PHẦN THỰC HÀNH	48
1. Bài tập 1:	48
2. Bài tập 2:	49
Bài 3: CÁC PHÉP TOÁN NHỊ PHÂN	50
I. Các lệnh vào/ra tiếp điểm	50
A. PHẦN LÝ THUYẾT:	50
1. Các lệnh vào:.....	50
2. Lệnh ra:	51
3. Bit nhớ trung gian:	52
B. PHẦN THỰC HÀNH:	52
4. Bài tập áp dụng:	52
C. PHẦN MỞ RỘNG:	53
II. Các liên kết nhị phân, đại số boolean.	54
A. PHẦN LÝ THUYẾT:	54
1. Phép toán AND:	54
2. Phép toán OR:	54
B. PHẦN THỰC HÀNH	55
3. Bài tập áp dụng:	55
III. Lệnh Set/Reset	55
A. PHẦN LÝ THUYẾT:	55
1. Lệnh Set(S).....	55
2. Lệnh Reset(S).....	55
3. Lệnh Set/Reset một FlipFlop:.....	56
B. PHẦN THỰC HÀNH	57
4. Bài tập áp dụng:	57
IV. Lệnh nhận biết cạnh tín hiệu:	57
A. PHẦN LÝ THUYẾT:	57
1. Nhận biết tín hiệu cạnh lên (Vi phân cạnh lên) :.....	57
2. Nhận biết tín hiệu cạnh xuống (Vi phân cạnh xuống):	57
3. Lệnh Save :.....	58
B. PHẦN THỰC HÀNH:	58
4. Bài tập áp dụng:	58
V. Lệnh nhảy	59

A. PHẦN LÝ THUYẾT:	59
1. Lệnh nhảy không điều kiện:	59
2. Lệnh nhảy có điều kiện.	60
B. PHẦN THỰC HÀNH:	60
3. Bài tập ứng dụng	60
Bài 4: CÁC PHÉP TOÁN SỐ HỌC	61
I. Các lệnh Timer:	61
A. PHẦN LÝ THUYẾT	61
1. Timer trong PLC:	61
2. Timer tạo xung không có nhớ (Pulse Timer – SP):	62
3. Timer tạo xung có nhớ (Extended Pulse Timer - SE)	62
4. Trễ theo sườn lên không có nhớ (On Delay Timer - SD)	63
5. Trễ theo sườn lên có nhớ (Latching On Delay Timer - SS)	63
6. Timer trễ theo sườn xuống (Off Delay Timer - SF)	64
B. PHẦN THỰC HÀNH:	65
7. Bài tập ứng dụng	65
II. Các lệnh Counter:	66
A. PHẦN LÝ THUYẾT	66
1. Nguyên lý làm việc của counter:	66
2. Khai báo sử dụng:	67
3. Các loại bộ đếm trong PLC S7-300: * UP COUNTER:	68
B. PHẦN THỰC HÀNH	70
4. Bài tập ứng dụng:	70
III. Lệnh nạp và truyền dữ liệu:	71
A. PHẦN LÝ THUYẾT:	71
Hình 4.7: Cấu trúc lệnh move	71
B. PHẦN THỰC HÀNH:	72
IV. Các lệnh so sánh:	72
A. PHẦN LÝ THUYẾT	72
B. PHẦN THỰC HÀNH:	73
V. Các lệnh chuyển đổi dữ liệu:	74
VII. Bài tập áp dụng các phép toán số học:	76
Bài 5: KẾT NỐI HỆ THỐNG ĐIỀU KHIỂN PLC	78

I. Giới thiệu phần cứng kit thí nghiệm S7 – 300:	78
A. PHẦN LÝ THUYẾT:	78
1.Module CPU:	78
2.Module mở rộng ngõ vào số (Digital Input Module (DI))	79
SM 321 DI 16 x DC24V/321 – 1BH02-0AA0):	79
3. Module mở rộng ngõ ra số (Digital Output Module (DO))	79
4. Analog Input Module SM 331:	79
5. Analog Output Module SM 332:	79
6. Các khối phụ trợ cho thí nghiệm	79
B. PHẦN THỰC HÀNH:	80
II. Cách kết nối dây cho hệ thống:	81
A. PHẦN LÝ THUYẾT:	81
1. Vẽ sơ đồ kết nối:	81
B. PHẦN THỰC HÀNH:	81
III. Bài tập ứng dụng:	81
1. Bài tập 1:	81
Hình 6.2b: Sơ đồ kết nối phần cứng	82
2. Bài tập 2	82
TÀI LIỆU THAM KHẢO	84

CHƯƠNG TRÌNH MÔ ĐUN PLC CƠ BẢN

Tên mô đun: PLC CƠ BẢN

Mã mô đun: MĐ 23

Thời gian thực hiện mô đun: 75 giờ (Lý thuyết: 15 giờ, thực hành, thí nghiệm, thảo luận: 56 giờ, kiểm tra: 4 giờ).

I. VỊ TRÍ, TÍNH CHẤT CỦA MÔ ĐUN

1. Vị trí: Trước khi học mô đun này cần hoàn thành các môn học, mô đun cơ sở và các môn học, mô đun chuyên môn, mô đun này nên học cuối cùng trong khóa học.

2. Tính chất: Là mô đun kỹ thuật chuyên ngành, thuộc các mô đun đào tạo nghề bắt buộc.

II. MỤC TIÊU CỦA MÔ ĐUN

1. Về kiến thức:

- Trình bày được nguyên lý hệ điều khiển lập trình PLC; So sánh các ưu nhược điểm với bộ điều khiển có tiếp điểm và các bộ lập trình cỡ nhỏ khác.

- Phân tích được cấu tạo phần cứng và nguyên tắc hoạt động của phần mềm trong hệ điều khiển lập trình PLC.

2. Về kỹ năng:

Viết thành thạo các chương trình điều khiển thông dụng trong công nghiệp, biết cách mô phỏng chương trình này trên phần mềm, Download được chương trình này về mô hình phần cứng, mô hình hoạt động đúng.

3. Về năng lực tự chủ và trách nhiệm:

- Trong quá trình làm việc luôn đảm bảo tính cẩn thận, chính xác, đúng qui trình, đảm bảo an toàn cho người và thiết bị.

Bài 1: ĐẠI CƯƠNG VỀ ĐIỀU KHIỂN LẬP TRÌNH

Mục tiêu của bài:

- Trình bày được các ưu điểm của điều khiển lập trình so với các loại điều khiển khác và các ứng dụng của chúng trong thực tế.
- Trình bày được cấu trúc và nhiệm vụ các khối chức năng của PLC.
- Truy cập được các địa chỉ các miền nhớ trong PLC.

Nội dung của bài:

I. Cấu trúc cơ bản của hệ thống điều khiển.

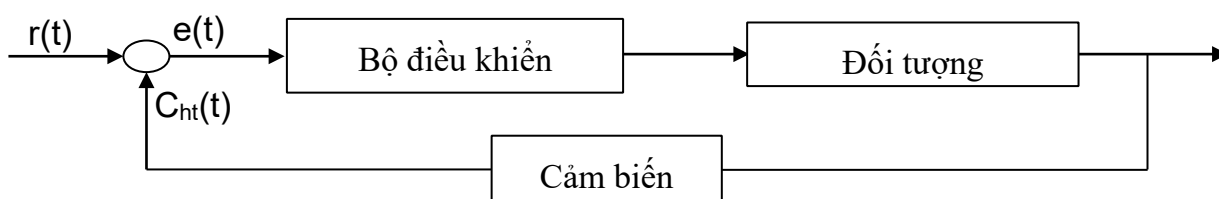
1. Tổng quan về các hệ thống điều khiển.

Do yêu cầu tự động hóa trong công nghiệp ngày càng tăng, đòi hỏi sự đáp ứng của kỹ thuật điều khiển. Để giải quyết vấn đề này, ta có hai cách: Dùng role, khởi động từ ... hoặc dùng chương trình có nhớ.

Hệ điều khiển bằng role và hệ điều khiển bằng lập trình có nhớ khác nhau chủ yếu ở phần xử lý: Thay vì dùng role, tiếp điểm và dây nối thì trong phương pháp lập trình có nhớ, chúng được thay bằng các mạch điện tử.

Ở hệ thống điều khiển bằng role điện, khi thay đổi nhiệm vụ điều khiển người ta phải thay đổi mạch điều khiển bằng cách lắp lại mạch, thay đổi phần tử mới. Trong khi đó, đối với hệ thống điều khiển bằng lập trình có nhớ, khi thay đổi nhiệm vụ điều khiển ta chỉ cần thay đổi chương trình soạn thảo.

Một hệ thống điều khiển thường luôn có ba thành phần cơ bản là ngõ vào, bộ điều khiển và tín hiệu ra, thường có sơ đồ khối như sau:

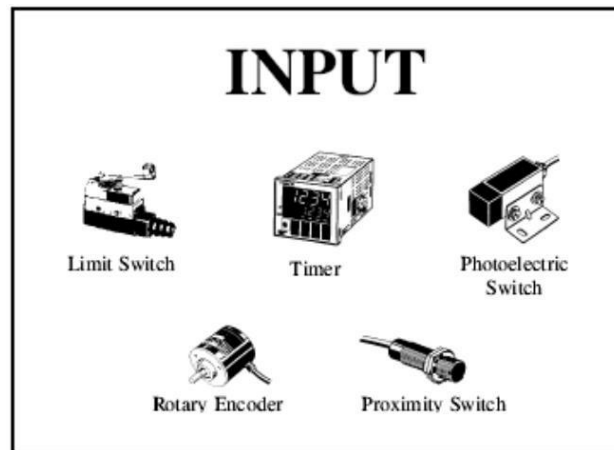


Hình 1.1: Sơ đồ khối của một hệ thống điều khiển

- $r(t)$: Ngõ vào chuẩn.
- $C_{ht}(t)$: Ngõ vào từ cảm biến.
- Đối tượng: Cơ cấu chấp hành.
- Bộ điều khiển: Chứa chương trình điều khiển hệ thống
- Cảm biến: Lấy tín hiệu đo cho bộ điều khiển.

2. Tín hiệu vào:

Tín hiệu vào ($r(t)$, $C_{ht}(t)$) được nhận từ nút nhấn, bàn phím và các chuyển mạch. Mặt khác, để đo, kiểm tra chuyển động, áp suất, lưu lượng chất lỏng ... PLC phải nhận tín hiệu từ các cảm biến. Ví dụ : tiếp điểm hành trình, cảm biến quang điện... tín hiệu đưa vào PLC có thể là tín hiệu số (Digital) hoặc tín hiệu tương tự (Analog). Các tín hiệu này được giao tiếp với PLC thông qua các Module nhận tín hiệu vào khác nhau DI (ngõ vào số) hoặc AI (ngõ vào tương tự)... Hình dạng một số thiết bị nhận tín hiệu vào được trình bày trong hình 1.2.



Hình 1.2: Một số thiết bị nhận tín hiệu vào

3. Bộ điều khiển.

Trong thực tế, bộ điều khiển có rất nhiều loại, đối với hệ thống điều khiển không dùng chương trình có nhớ thì bộ điều khiển là các côngtactơ, các loại role. Đối với hệ thống điều khiển dùng chương trình có nhớ, ta có các bộ điều khiển thông dụng sau :

a) Máy tính :

- Được dùng trong những chương trình phức tạp, đòi hỏi độ chính xác cao. -
Cáo giao diện thân thiện.

- Tốc độ xử lý cao.

- Có thể lưu trữ chương trình và dữ liệu với dung lượng lớn **b) Vi xử lý :**

- Được dùng trong những chương trình có độ phức tạp không cao (vì chỉ xử lý được 8 bits), độ chính xác thấp.

- Giao diện không thân thiện.

- Tốc độ xử lý không cao.

- Không lưu trữ hoặc chỉ lưu trữ với dung lượng rất ít.

- Không bền trong môi trường công nghiệp.

- Giá thành thấp

c) Bộ điều khiển bằng PLC: - Bền trong môi trường công nghiệp.

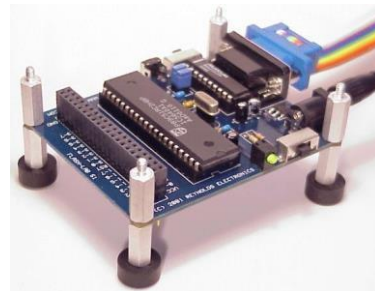
- Giao diện không thân thiện với người sử dụng. Phải kết nối giao diện với máy tính tay Touchscreen..

- Tốc độ xử lý tương đối cao

- Có nhiều loại khác nhau để lựa chọn tùy nhu cầu sử dụng và độ phức tạp của hệ thống điều khiển.



PLC (Programmable Logic Control)



Vi xử lý

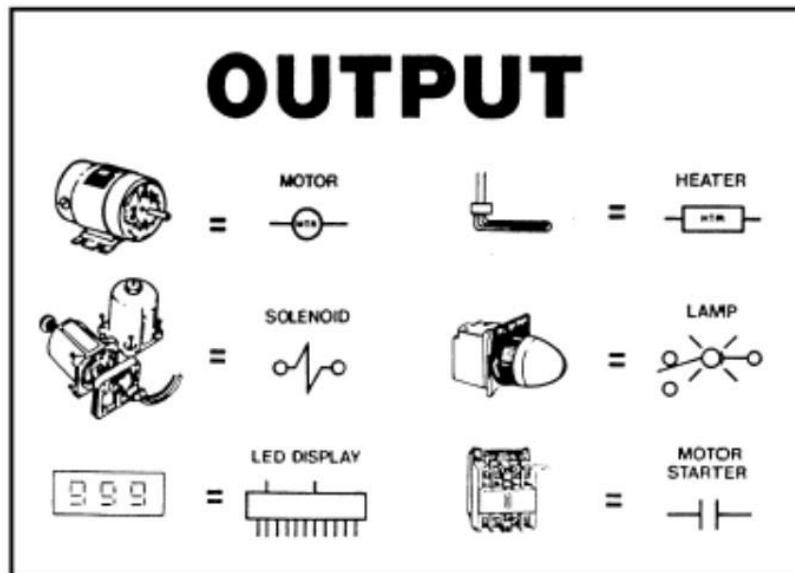


Giao tiếp máy tính

Hình 1.3 : Các bộ điều khiển.

4. Tín hiệu ra.

Một hệ thống điều khiển sẽ không có ý nghĩa thực tế nếu không giao tiếp được với thiết bị xuất, các thiết bị xuất thông dụng như: Mô tơ, van, xylanh, role, đèn báo, động cơ, chuông điện... Cũng như thiết bị nhập, các thiết bị xuất được nối vào các ngõ ra của Module ra (Output). Các Module ra này có thể là DO (ngõ ra số) hoặc AO (ngõ ra tương tự).



Hình 1.4 : Các thiết bị đầu ra

II. Giới thiệu về PLC

1. Sơ lược về lịch sử phát triển:

Thiết bị điều khiển lập trình đầu tiên được hình thành từ một nhóm kỹ sư của hãng General Motors vào năm 1968 với ý tưởng ban đầu là thiết kế một bộ điều khiển thỏa mãn các yêu cầu sau:

- Lập trình dễ dàng, ngôn ngữ lập trình dễ hiểu.
- Dễ dàng sửa chữa, thay thế.
- Ổn định trong môi trường công nghiệp.
- Giá cả cạnh tranh.

Tuy nhiên, ban đầu khi mới ra đời, hệ thống này còn khá đơn giản và cồng kềnh, người sử dụng gặp nhiều khó khăn trong việc vận hành hệ thống. Vì vậy các nhà thiết kế từng bước cải tiến hệ thống đơn giản, gọn nhẹ, dễ vận hành, nhưng việc lập trình cho hệ thống còn khó khăn, do lúc này không có các thiết bị lập trình ngoại vi hỗ trợ cho công việc lập trình.

Để đơn giản hóa việc lập trình, hệ thống điều khiển lập trình cầm tay (programmable controller handle) đầu tiên được ra đời vào năm 1969. Điều này đã tạo ra một sự phát triển thật sự cho kỹ thuật điều khiển lập trình. Trong giai đoạn này các hệ thống điều khiển lập trình (PLC) chỉ đơn giản nhằm thay thế hệ thống Relay và dây nối trong hệ thống điều khiển cổ điển. Qua quá trình vận hành, các nhà thiết kế đã từng bước tạo ra được một tiêu chuẩn mới cho hệ thống, tiêu chuẩn đó là :Dạng lập trình dùng giản đồ hình thang (The ladder format). Trong những năm đầu thập niên 1970, những hệ thống PLC còn có thêm khả năng vận hành với những thuật toán hỗ trợ (arithmetic), “vận hành với các dữ liệu cập nhật” (data manipulation). Do sự phát triển của loại màn hình dùng cho máy tính (Cathode Ray

Tube: CRT), nên việc giao tiếp giữa người điều khiển để lập trình cho hệ thống càng trở nên thuận tiện hơn.

Sự phát triển của hệ thống phần cứng và phần mềm từ năm 1975 cho đến nay đã làm cho hệ thống PLC phát triển mạnh mẽ hơn với các chức năng mở rộng: hệ thống ngõ vào/ra có thể tăng lên đến 8.000 cổng vào/ra, dung lượng bộ nhớ chương trình tăng lên hơn 128.000 từ bộ nhớ (word of memory). Ngoài ra các nhà thiết kế còn tạo ra kỹ thuật kết nối với các hệ thống PLC riêng lẻ thành một hệ thống PLC chung, tăng khả năng của từng hệ thống riêng lẻ. Tốc độ xử lý của hệ thống được cải thiện, chu kỳ quét (scan) nhanh hơn làm cho hệ thống PLC xử lý tốt với những chức năng phức tạp số lượng cổng ra/vào lớn.

Trong tương lai hệ thống PLC không chỉ giao tiếp với các hệ thống khác thông qua CIM (Computer Intergrated Manufacturing) để điều khiển các hệ thống: Robot, Cad/Cam... ngoài ra các nhà thiết kế còn đang xây dựng các loại PLC với các chức năng điều khiển “thông minh” (intelligence) còn gọi là các siêu PLC (super PLCs) cho tương lai.

2. Cấu trúc cơ bản và hoạt động của một PLC

a) Cấu trúc:

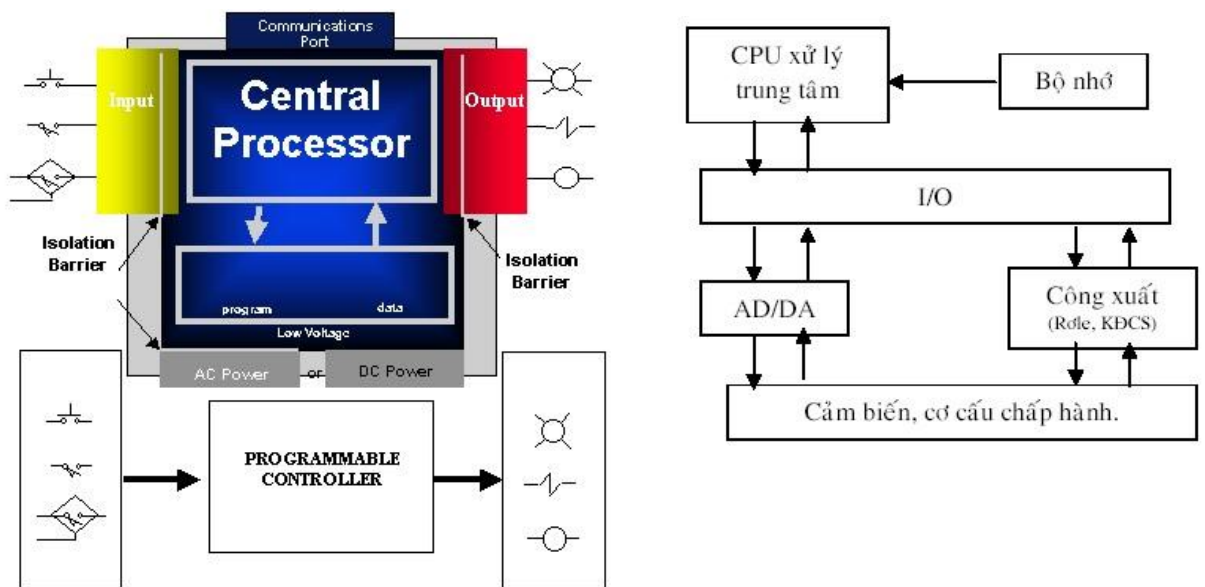
* Đặc điểm cơ bản của một PLC:

- Là loại thiết bị cho phép thực hiện linh hoạt các thuật toán điều khiển số thông qua một ngôn ngữ lập trình, thay cho việc thể hiện thuật toán đó bằng mạch số.

- Như vậy PLC là bộ điều khiển số nhỏ gọn, dễ dàng thay đổi thuật toán, dễ trao đổi thông tin với máy tính và các PLC khác.

- Chương trình điều khiển được lưu trữ toàn bộ trong bộ nhớ dưới dạng các khối và được thực hiện lặp lại liên tục theo chu kỳ quét.

- Vì là bộ điều khiển nên PLC cũng có tính năng như một máy tính với: Bộ phận xử lý trung tâm (CPU: Central Processing Unit) và các ngõ vào ra. Bộ phận xử lý trung tâm (CPU) gồm ba phần: bộ xử lý, hệ thống bộ nhớ và hệ thống nguồn cung cấp. Hình 1.4 mô tả ba phần cấu thành một PLC:



Hình 1.5: Sơ đồ khối một PLC

Từ sơ đồ khối trên, ta thấy PLC được chia làm 3 thành phần rõ rệt với nhiệm vụ cụ thể như sau:

- Các ngõ vào: nhận tín hiệu vào từ bàn phím, các cảm biến... đa phần đây là các tín hiệu xuất phát từ tác động của người vận hành hoặc đã được người vận hành hệ thống cài đặt trước, các tín hiệu này thể hiện yêu cầu cần đáp ứng của hệ thống, giúp cho hệ thống hoạt động y theo ý muốn người vận hành.

- Bộ phận xử lý trung tâm: Bộ phận này có chức năng xử lý tín hiệu vào theo chương trình đã cài đặt và lưu trữ chương trình, lưu các dữ liệu cần thiết, sau đó đưa kết quả xử lý đến ngõ ra. Đây là bộ phận quan trọng và phức tạp nhất trong hệ thống. Nếu các tín hiệu vào/ra là tín hiệu tương tự thì nó còn bao gồm bộ phận biến đổi AD/DA.

- Các ngõ ra: Đưa tín hiệu ra đến các thiết bị cần điều khiển như các đèn báo, chuông báo, motor... Đây là kết quả mà người vận hành hệ thống muốn có được khi tác động đầu vào.

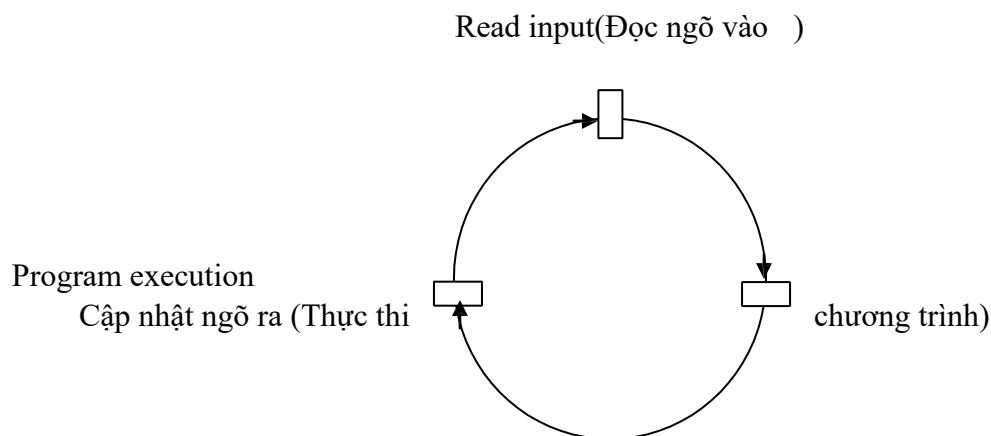
b) Hoạt động của một PLC:

Về cơ bản hoạt động của một PLC cũng khá đơn giản: Đầu tiên, hệ thống các cổng vào/ra (Input/Output) (còn gọi là các Module xuất /nhập) dùng để nhận các tín hiệu từ các thiết bị ngoại vi đưa vào CPU (như các sensor, công tắc, tín hiệu từ động cơ ...). Sau khi nhận được tín hiệu ở ngõ vào thì CPU sẽ xử lý và đưa các tín hiệu điều khiển qua Module xuất ra các thiết bị cần điều khiển.

Trong suốt quá trình hoạt động, CPU đọc hoặc quét (scan) dữ liệu hoặc trạng thái của thiết bị ngoại vi thông qua ngõ vào, sau đó thực hiện các chương trình trong

bộ nhớ như sau: một bộ đếm chương trình sẽ nhận lệnh từ bộ nhớ chương trình đưa ra thanh ghi lệnh để thi hành. Chương trình ở dạng STL (StatementList – Dạng lệnh liệt kê) sẽ được dịch ra ngôn ngữ máy cất trong bộ nhớ chương trình. Sau khi thực hiện xong chương trình, CPU sẽ gửi hoặc cập nhật (Update) tín hiệu tới các thiết bị, được thực hiện thông qua module xuất. Một chu kỳ gồm đọc tín hiệu ở ngõ vào, thực hiện chương trình và gửi cập nhật tín hiệu ở ngõ ra được gọi là một chu kỳ quét (Scanning).

Trên đây chỉ là mô tả hoạt động đơn giản của một PLC, với hoạt động này sẽ giúp cho người thiết kế nắm được nguyên tắc của một PLC. Nhằm cụ thể hóa hoạt động của một PLC, sơ đồ hoạt động của một PLC là một vòng quét (Scan) như sau:



Hình 1.6 :Một vòng quét của PLC.

Thường việc thực thi một vòng quét xảy ra với một thời gian rất ngắn, một vòng quét đơn (single scan) có thời gian thực hiện từ 1ms tới 100ms. Việc thực hiện một chu kỳ quét dài hay ngắn còn phụ thuộc vào độ dài của chương trình và cả mức độ giao tiếp giữa PLC với các thiết bị ngoại vi (màn hình hiển thị...). Thông thường vi xử lý có thể đọc được mọi tín hiệu ở ngõ vào, chỉ khi nào tín hiệu này tác động với khoảng thời gian nhỏ hơn một chu kỳ quét thì vi xử lý coi như không có tín hiệu này. Tuy nhiên trong thực tế sản xuất, thường các hệ thống chấp hành là các hệ thống cơ khí nên có tốc độ quét như trên có thể đáp ứng được các chức năng của dây chuyền sản xuất. Để khắc phục thời gian quét dài, ảnh hưởng đến chu trình sản xuất các nhà thiết kế còn thiết kế hệ thống PLC cập nhật tức thời,

các hệ thống này thường được áp dụng cho các PLC lớn có số lượng I/O nhiều, truy cập và xử lý lượng thông tin lớn.

c) Phân loại PLC:

PLC được phân loại theo ba cách:

- Phân loại theo hãng sản xuất: gồm có các nhãn hiệu như: Siemens, Omron, Misubishi,...

- Phân loại theo họ (Version):

- + PLC Siemens có các họ S7-200, S7-300, S7-400, Logo...

- + Misubishi có các họ: FX, FX₀, FX_{0N},...

- Phân loại theo khả năng làm việc: loại nhỏ, loại trung bình và loại lớn.

Đầu tiên là khả năng và giá trị cũng như nhu cầu về hệ thống sẽ giúp người sử dụng chọn những loại PLC phù hợp với nhu cầu sử dụng. Nhu cầu về hệ thống được xem như là một nhu cầu ưu tiên nó giúp người sử dụng biết cần loại PLC nào và đặc trưng của từng loại để dễ dàng lựa chọn.

Hình 1.7 cho ta cách phân loại theo kiểu “bậc thang” và việc sử dụng PLC sao cho phù hợp với các hệ thống thực tế. Trong hình này ta có thể nhận thấy những vùng chồng lên nhau, ở những vùng này người sử dụng thường phải sử dụng các loại PLC đặc biệt như: số lượng cổng vào/ra (I/O) có thể sử dụng ở vùng có số I/O thấp nhưng lại có các tính năng đặc biệt của các PLC ở vùng có số lượng I/O cao. Thường người sử dụng các loại PLC thuộc vùng chồng lấn nhằm tăng tính năng của PLC đồng thời lại giảm thiểu số lượng I/O không cần thiết.

Các nhà thiết kế phân PLC ra thành các loại sau:

*** Loại 1 : Micro PLC (PLC siêu nhỏ)**

Micro PLC thường được ứng dụng trong các dây chuyền sản xuất nhỏ, các ứng dụng trực tiếp trong từng thiết bị đơn lẻ (ví dụ: điều khiển băng tải nhỏ). Các PLC này thường được lập trình bằng các bộ lập trình cầm tay, một vài micro PLC còn có khả năng hoạt động với tín hiệu I/O tương tự (analog) (ví dụ: việc điều khiển nhiệt độ). Các tiêu chuẩn của một Micro PLC như sau: - 32 ngõ vào/ra.

- Sử dụng vi xử lý 8 bit.

- Thường dùng thay thế role.

- Bộ nhớ có dung lượng 1K.

- Ngõ vào/ra là tín hiệu số. - Có timers và counters.

- Thường được lập trình bằng các bộ lập trình cầm tay.

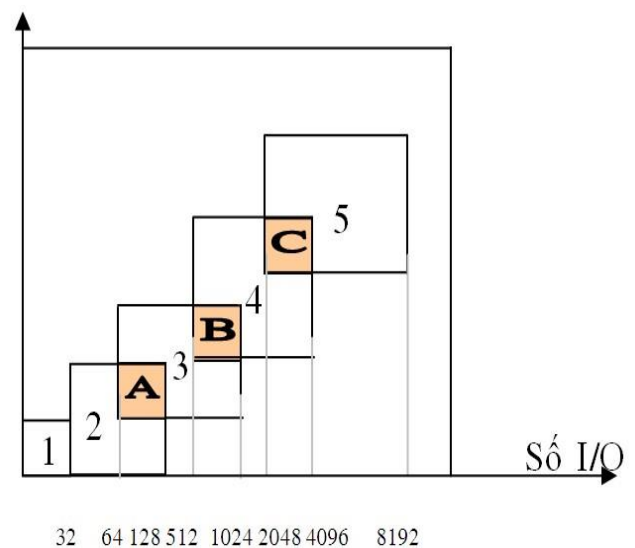
*** Loại 2: PLC cỡ nhỏ (Small PLC)**

Small PLC thường được dùng trong việc điều khiển các hệ thống nhỏ (ví dụ : Điều khiển động cơ, dây chuyền sản xuất nhỏ), chức năng của các PLC này thường

được giới hạn trong việc thực hiện chuỗi các mức logic, điều khiển thay thế role. Các tiêu chuẩn của một small PLC như sau:

- Có 128 ngõ vào/ra (I/O).
- Dùng vi xử lý 8 bit.
- Thường dùng để thay thế các role.
- Dùng bộ nhớ 2K.
- Lập trình bằng ngôn ngữ dạng hình thang (ladder) hoặc liệt kê.
- Có timers/counters/thanh ghi dịch (shift registers).
- Đồng hồ thời gian thực.
- Thường được lập trình bằng bộ lập trình cầm tay.

Chú ý vùng A trong sơ đồ hình 1.7. Ở đây dùng PLC nhỏ với các chức năng tăng cường của PLC cỡ lớn hơn như: Thực hiện được các thuật toán cơ bản, có thể nối mạng, cổng vào ra có thể sử dụng tín hiệu tương tự.



Hình 1.7 : Cách dùng các loại PLC.

* Loại 3: PLC cỡ trung bình (Medium PLCs)

PLC trung bình có hơn 128 đường vào/ra, điều khiển được các tín hiệu tương tự, xuất nhập dữ liệu, ứng dụng được những thuật toán, thay đổi được các đặc tính của PLC nhờ vào hoạt động của phần cứng và phần mềm (nhất là phần mềm) các thông số của PLC trung bình như sau:

- Có khoảng 1024 ngõ vào/ra (I/O).
- Dùng vi xử lý 8 bit.
- Thay thế role và điều khiển được tín hiệu tương tự.
- Bộ nhớ 4K, có thể nâng lên 8K.
- Tín hiệu ngõ vào ra là tương tự hoặc số.
- Có các lệnh dạng khối và ngôn ngữ lập trình là ngôn ngữ cấp cao. - Có timers/Counters/Shift Register.

- Có khả năng xử lý chương trình con (qua lệnh JUMP...).
- Có các lệnh dạng khối và ngôn ngữ lập trình là ngôn ngữ cấp cao.
- Thực hiện các thuật toán (cộng, trừ, nhân, chia...).
- Giới hạn dữ liệu với bộ lập trình cầm tay.
- Có đường tín hiệu đặc biệt ở module vào/ra.
- Giao tiếp với các thiết bị khác qua cổng RS232.
- Có khả năng hoạt động với mạng.
- Lập trình qua CRT (Cathode Ray Tube) để dễ quan sát.

Chú ý tới vùng B (hình 1.6) PLC ở vùng B thường trực được dùng do có nhiều bộ nhớ hơn, điều khiển mạng PID có khả năng thực hiện những chuỗi lệnh phần lớn về thuật toán hoặc quản lý dữ liệu.

* Loại 4: PLC cỡ lớn (large PLC)

Large PLC được sử dụng rộng rãi hơn do có khả năng hoạt động hữu hiệu, có thể nhận dữ liệu, báo những dữ liệu đã nhận... Phần mềm cho thiết bị điều khiển cầm tay được phát triển mạnh hơn tạo thuận lợi cho người sử dụng. Tiêu chuẩn PLC cỡ lớn: Ngoài các tiêu chuẩn như PLC cỡ trung, PLC cỡ lớn còn có thêm các tiêu chuẩn sau:

- Có 2048 cổng vào/ra (I/O).
- Dùng vi xử lý 8 bit hoặc 16 bit.
- Bộ nhớ cơ bản có dung lượng 12K, mở rộng lên được 32K. - Local và remote I/O.
- Điều khiển hệ thống role (MCR: Master Control Relay).
- Chuỗi lệnh, cho phép ngắt (Interrupts).
- PID hoặc làm việc với hệ thống phần mềm PID.
- Hai hoặc nhiều hơn cổng giao tiếp RS 232.
- Nối mạng.
- Dữ liệu điều khiển mở rộng, so sánh, chuyển đổi dữ liệu, chức năng giải thuật toán mã điều khiển mở rộng (mã nhị phân, hexa ...). - Có khả năng giao tiếp giữa máy tính và các module.

* Loại 5: PLC rất lớn (very large PLCs)

Very large PLC được dùng trong các ứng dụng đòi hỏi sự phức tạp và chính xác cao, đồng thời dung lượng chương trình lớn. Ngoài ra PLC loại này còn có thể giao tiếp I/O với các chức năng đặc biệt, tiêu chuẩn PLC loại này ngoài các chức năng như PLC loại lớn còn có thêm các chức năng:

- Có 8192 cổng vào/ra (I/O).
- Dùng vi xử lý 16 bit hoặc 32 bit.
- Bộ nhớ 64K, mở rộng lên được 1M.

- Thuật toán :+, -, *, /, bình phương.
- Dữ liệu điều khiển mở rộng : Bảng mã ASCII, LIFO, FIFO.

d) So sánh PLC với các hệ thống điều khiển khác:

*** So sánh PLC với hệ thống điều khiển bằng role:**

Việc phát triển hệ thống điều khiển bằng lập trình đã dần thay thế từng bước hệ thống điều khiển bằng role trong các quá trình sản xuất. Khi thiết kế một hệ thống điều khiển hiện đại, người kỹ sư phải cân nhắc, lựa chọn giữa các hệ thống điều khiển lập trình và hệ thống điều khiển bằng role. Hệ thống điều khiển lập trình thường được sử dụng do các ưu điểm sau:

- Hệ thống điều khiển có độ mềm dẻo cao, chỉ cần lắp đặt một lần (đối với sơ đồ hệ thống, các đường nối dây, các tính hiệu ở ngõ vào/ra ...), mà không phải thay đổi kết cấu của hệ thống sau này, giảm được sự tốn kém khi phải thay đổi lắp đặt khi đổi thứ tự điều khiển (đối với hệ thống điều khiển relay ...), khả năng chuyển đổi hệ điều khiển cao hơn (như giao tiếp giữa các PLC để truyền dữ liệu điều khiển lẫn nhau), hệ thống được điều khiển linh hoạt hơn. Khi cần thay đổi hay mở rộng hệ thống thì chủ yếu là thay đổi phần lập trình.

- PLC chiếm một khoảng không gian nhỏ hơn nhưng điều khiển nhanh, nhiều chức năng hơn các hệ thống khác, nhất là đối với các hệ thống điều khiển lớn, phức tạp, và quá trình lắp đặt hệ thống PLC ít tốn thời gian so với các hệ thống khác.

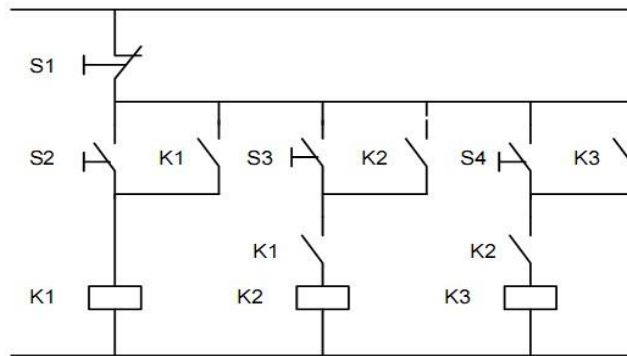
- Việc lắp đặt sơ đồ điều khiển cũng đơn giản hơn hệ thống điều khiển cổ điển

(dùng role)

- Tốc độ xử lý thời gian thực tương đối cao.
- Công suất tiêu thụ nhỏ.
- Có khả năng mở rộng số ngõ vào/ra khi nhu cầu điều khiển tăng lên, chỉ bằng cách nối thêm các module vào/ra chức năng.
- Dễ dàng điều khiển và giám sát từ máy tính: người sử dụng có thể nhận biết các trục trặc hệ thống của PLC nhờ giao diện qua màn hình máy tính (một số PLC thế hệ sau có khả năng nhận biết các hỏng hóc (trouble shoding) của hệ thống và báo cho người sử dụng), điều này làm cho việc sửa chữa thuận lợi hơn.
- Giá thành hợp lý tùy vào từng loại PLC.

Thí dụ sau cho ta thấy ưu điểm cơ bản của PLC so với hệ thống điều khiển bằng role: Điều khiển một hệ thống 3 máy bơm nước bằng 3 khởi động từ K1, K2, K3. Trình tự điều khiển như sau: các máy bơm hoạt động theo trình tự nhất định là K1

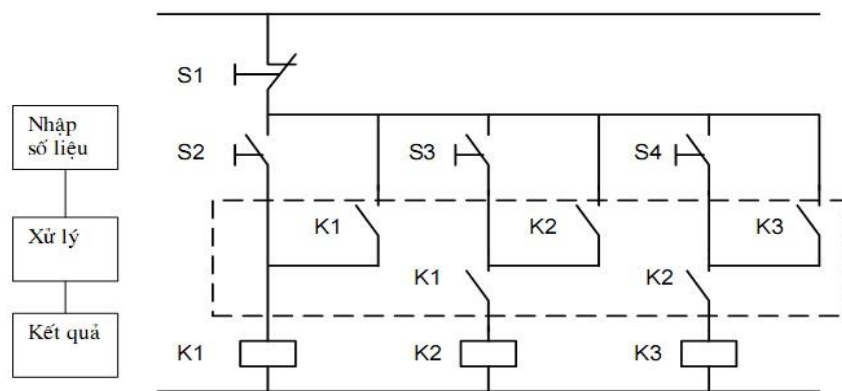
đóng trước, tiếp đến là K2 và cuối cùng là K3. Ta có sơ đồ điều khiển bằng role như sau:



Hình 1.8a: Sơ đồ điều khiển bằng role

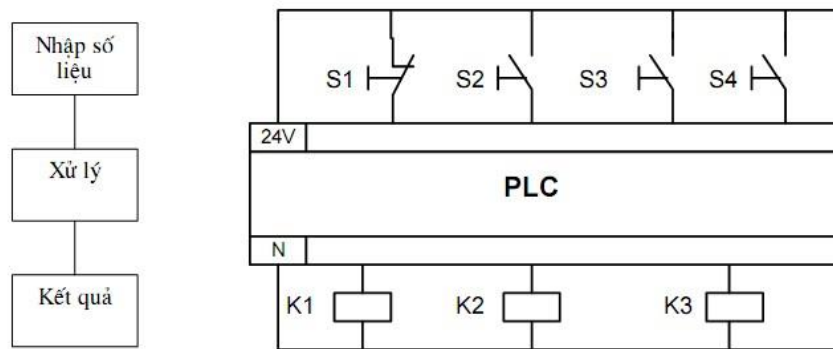
Nếu thay bằng thiết bị điều khiển PLC, ta có thể mô tả như sau:

- Tín hiệu vào là các nút nhấn S1, S2, S3, S4 vẫn giữ nguyên.
- Tín hiệu ra: K1, K2, K3 là các khởi động từ vẫn giữ nguyên.
- Bộ phận xử lý được thay thế bằng PLC



Hình 1.8b: Sơ đồ khối tương đương của hình 1.8a.

Khi thực hiện bằng chương trình điều khiển có nhớ PLC, ta chỉ cần kết nối theo sơ đồ sau:



Hình 1.9: Sơ đồ kết nối dùng PLC

Nếu bây giờ nhiệm vụ điều khiển thay đổi, thí dụ như các bơm nước 1, 2, 3 hoạt động theo trình tự ngược lại hoặc các bơm nước hoạt động độc lập thì đối với hệ thống dùng role, ta phải lắp lại mạch, còn đối với hệ thống PLC ta chỉ thay đổi phân lập trình mà không thay đổi bất cứ kết nối phần cứng nào.

* So sánh PLC với máy tính:

PLC và máy tính đều dựa trên bộ xử lý trong tâm (CPU) để xử lý dữ liệu. Tuy nhiên có một vài cấu trúc quan trọng cần phân biệt để thấy rõ sự khác biệt giữa một PLC và một máy tính.

- Không như một máy tính, PLC được thiết kế đặc biệt để hoạt động trong môi trường công nghiệp. Một PLC có thể được lắp đặt ở những nơi có độ nhiễu điện cao, vùng có từ trường mạnh, có các chấn động cơ khí, nhiệt độ môi trường cao ...

- PLC được thiết kế với phần cứng và phần mềm sao cho dễ lắp đặt (đối với phần cứng), đồng thời về phần mềm cũng phải dễ dàng để người sử dụng (kỹ sư, kỹ thuật viên) lập trình một cách nhanh chóng, thuận lợi (ví dụ: lập trình bằng ngôn ngữ hình thang ...).

- Máy tính không có các cổng giao tiếp trực tiếp với các thiết bị điều khiển, đồng thời máy tính cũng hoạt động không tốt trong môi trường công nghiệp.

- Ngôn ngữ lập trình trên máy tính không phải dạng hình thang, máy tính ngoài việc sử dụng các phần mềm chuyên biệt cho PLC, nó còn sử dụng các phần mềm khác nên làm “chậm” đi quá trình giao tiếp với các thiết bị được điều khiển.

Tuy nhiên qua máy tính, PLC có thể dễ dàng kết nối với các hệ thống khác, cũng như PLC có thể sử dụng bộ nhớ (có dung lượng rất lớn) của máy tính làm bộ nhớ của PLC.

e) Một vài lĩnh vực sử dụng PLC.

- Hóa học và dầu khí: định áp suất (dầu), bơm dầu, điều khiển hệ thống ống dẫn, cân đong trong ngành hóa ...

- Chế tạo máy và sản xuất: Tự động hoá trong chế tạo máy, quá trình lắp đặt máy, điều khiển nhiệt độ lò kim loại...
- Bột giấy, giấy, xử lý giấy. Điều khiển máy băm, quá trình cán, gia nhiệt ...
- Thủy tinh và phim ảnh: quá trình đóng gói, thí nghiệm vật liệu, các khâu hoàn tất sản phẩm, đo cắt giấy .
- Thực phẩm, rượu bia, thuốc lá: đếm sản phẩm, kiểm tra sản phẩm, kiểm soát quá trình sản xuất, bơm (bia, nước trái cây ...) cân đong, đóng gói, hòa trộn ...
- Kim loại: Điều khiển quá trình cán, cuộn (thép), qui trình sản xuất, kiểm tra chất lượng.
- Năng lượng: Điều khiển nguyên liệu (cho quá trình đốt, xử lý trong các turbin ...) các trạm cần hoạt động tuần tự khai thác vật liệu một cách tự động (than, gỗ, dầu mỏ).
- Hệ thống nâng vận chuyển, hệ thống báo động, điều khiển đèn giao thông. - Điều khiển các robot lắp ráp sản phẩm
- Quản lý bãi đậu xe tự động.

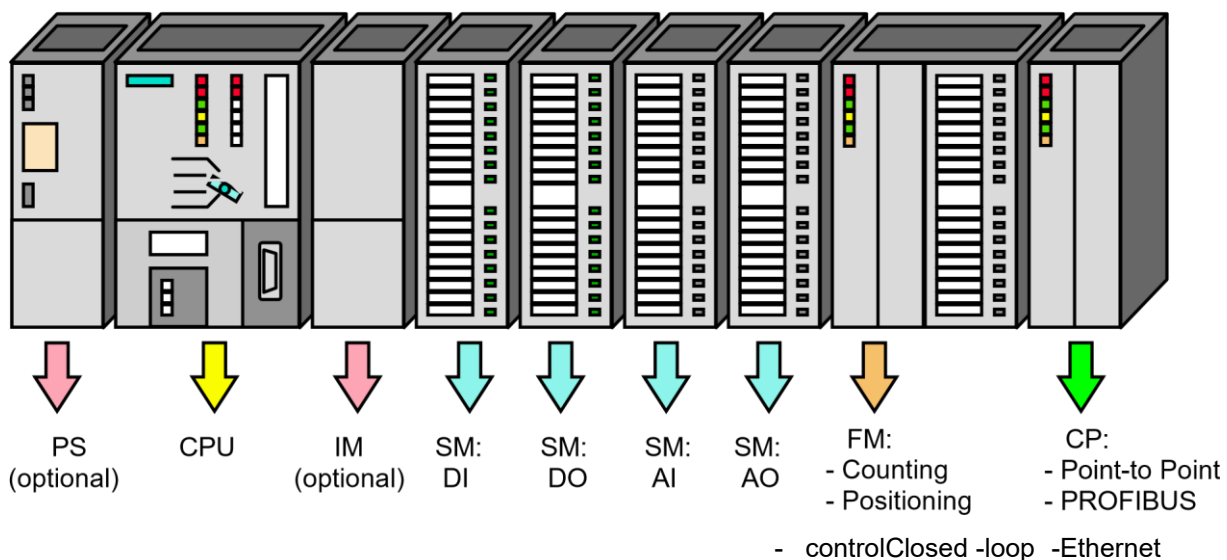
III. PLC S7-300

1. Các tính năng của PLC S7-300

- Hệ thống điều khiển theo kiểu module nhỏ gọn, sử dụng cho các ứng dụng có phạm vi trung bình.
- Có nhiều loại CPU khác nhau.
- Có nhiều Module mở rộng, có thể mở rộng đến 32 Module.
- Các bus nối tổ hợp phía sau các module.
- Có thể nối mạng MultiPoint Interface (MPI), ProfiBus hoặc Industrial Ethernet.
- Thiết bị lập trình trung tâm có thể kết nối đến từng Module.
- Không hạn chế số rãnh cắm.
- Có thể cài đặt thông số và cấu hình dễ dàng thông qua công cụ trợ giúp HWconfig.

2. Cấu trúc phần cứng PLC S7-300:

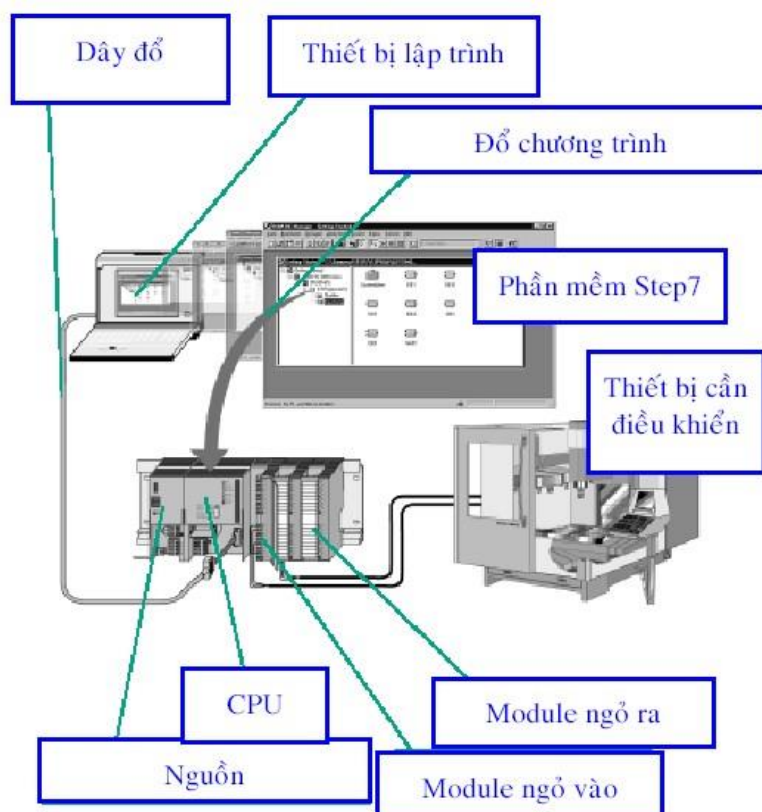
Cấu trúc phần cứng của một bộ PLC S7-300 thường gồm các bộ phận cơ bản sau:



Industrial

Hình 1.10: Sơ đồ kết nối đầy đủ các module trên rack

Tuy nhiên ngoài các module trên, một PLC muốn làm việc được thì trước tiên nó phải có thiết bị lập trình (vì ta không thể lập trình trực tiếp trên PLC) và để một bộ PLC có ý nghĩa thực tế thì ngõ ra của nó phải tác động đến các thiết bị cần điều khiển (chuông báo, motor...) nên ta có cấu trúc hoàn chỉnh của hệ thống điều khiển dùng PLC như sau:



Hình 1.11: Cấu trúc của PLC và các bộ phận phụ trợ






a) Module CPU:

- Module CPU chứa vi xử lý trung tâm, hệ điều hành, bộ nhớ, các bộ định thì, bộ đếm, cổng truyền thông (RS485), ... và có thể có vài cổng vào/ra số. Các cổng vào/ra số có trên CPU được gọi là cổng vào/ra onboard.

- Trong họ PLC S7-300 có nhiều loại CPU khác nhau, được đặt tên theo tên của bộ vi xử lý bên trong nó như CPU312, CPU312I, CPU314, CPU314FM,...

- Những module cùng sử dụng một loại bộ vi xử lý, nhưng khác nhau về cổng vào/ra onboard cũng như các khối hàm đặc biệt được tích hợp sẵn trong thư viện của hệ điều hành phục vụ việc sử dụng các cổng vào/ra onboard này sẽ phân biệt với nhau trong tên gọi bằng cách thêm cụm chữ cái IFM (Intergrated Function Module). Ví dụ như module CPU313IFM, CPU314IFM...

- Ngoài ra, còn có các loại module CPU với hai cổng truyền thông, trong đó cổng truyền thông thứ hai có chức năng chính là phục vụ kết nối mạng phân tán như mạng PROFIBUS (PROcess FIeld BUS). Tất nhiên, kèm theo cổng truyền thông thứ hai này là những phần mềm tiện dụng tích hợp cũng đã được cài sẵn trong hệ điều hành. Các loại module CPU này được phân biệt với các loại Module CPU khác bằng cách thêm cụm từ DP (Distributed Port). Ví dụ như Module CPU315-2DP. Tham khảo hình dưới đây:

CPU 312 IFM CPU 313 CPU 314 IFM CPU 314	CPU 315-2 DP CPU 316-2 DP		CPU 318-2	
MPI interface	MPI interface	PROFIBUS-DP interface	MPI/DP Interface	PROFIBUS-DP interface
				
-	-	-	Reconfiguration as a PROFIBUS-DP interface is possible	-

Hình 1.12: Cổng giao tiếp của các PLC

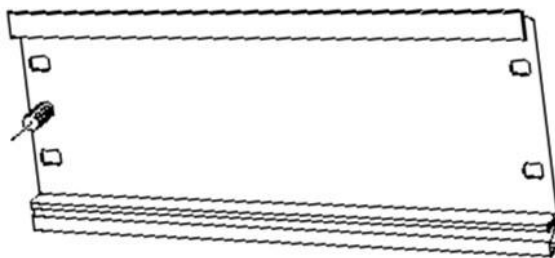
b) Các Module mở rộng:

* **PS (Power Supply)** Module cấp nguồn, có 3 loại 2A, 5A và 10A, đây là Module tùy chọn, có thể có hoặc không.

* **SM (Signal Module)** các module tín hiệu gồm có:

- DI (Digital Input): là các module mở rộng cổng vào số, có đặc điểm như sau:
- + Số lượng cổng có thể là 8, 16 hoặc 32 tùy theo từng loại module.

- + Tín hiệu vào là 24VDC.
- DO (Digital Output): là các module mở rộng cổng ra số, có đặc điểm như sau:
 - + Số lượng cổng có thể là 8, 16 hoặc 32 tùy theo từng loại module.
 - + Là các ngắt điện từ.
- DI/DO (Digital Input/ Digital Output): là các module mở rộng cổng vào/ra số, số lượng cổng có thể là 8vào/8ra hoặc 16vào/16ra tùy từng loại module.
- AI (Analog Input): là các module mở rộng cổng vào tương tự, có đặc điểm như sau:
 - + Là những bộ chuyển đổi tương tự sang số 12 bits (A/D)
 - + Số các cổng vào tương tự có thể là 2, 4 hoặc 8 tùy theo loại module + Tín hiệu vào có thể là áp, dòng, điện trở.
- AO (Analog Output): là các module mở rộng cổng ra tương tự, có đặc điểm như sau:
 - + Là những bộ chuyển đổi số sang tương tự 12 bits (D/A)
 - + Số các cổng vào tương tự có thể là 2, 4 hoặc 8 tùy theo loại module + Tín hiệu ra có thể là áp hoặc dòng.
- AI/AO (Analog Input/ Analog Output): là các module mở rộng cổng vào/ra tương tự, số lượng cổng có thể là 4 vào/2 ra hoặc 4 vào/4 ra tùy từng loại module.
- * **IM (Interface Module – Module giao tiếp)**: Đây là module tùy chọn, được dùng để ghép nối các nhóm module mở rộng lại với nhau thành một khối và được quản lý chung bởi một CPU.
- Các module mở rộng được cố định chung trên một thanh rack và có thứ tự như hình 1.10



Hình 1.13: Cấu trúc thanh rack

- Mỗi thanh rack chứa tối đa 8 module mở rộng (không kể CPU và nguồn nuôi)
- Một module CPU S7-300 có thể làm việc trực tiếp với nhiều nhất là 4 rack và các rack này phải được kết nối với nhau bằng module IM như hình 1.14:
- * **FM (Function Module – Module chức năng)**: Module có chức năng điều khiển chuyên biệt:
 - Module điều khiển động cơ servo
 - Module điều khiển động cơ bước - Module PID
 - Module điều khiển vòng kín.

- Module định vị...

* **CP (Communication Module – Module truyền thông):** Module phục vụ xử lý truyền thông trong mạng:

- MPI (mạng giao tiếp điểm – điểm)

- Profibus (mạng phân tán)

- Industrial Ethernet (mạng truyền thông công nghiệp)



Hình 1.14: sơ đồ kết nối nhiều rack

3. Cấu trúc bộ nhớ PLC S7-300:

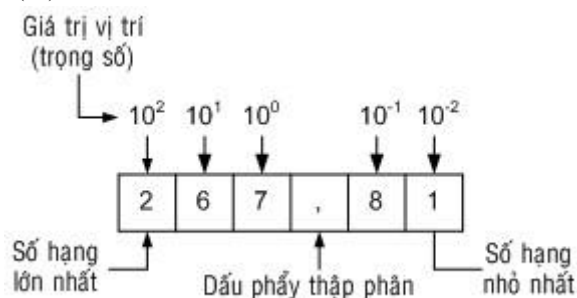
a) Kiến thức cơ sở: * Hệ thống số thập phân:

Trong hệ thập phân người ta sử dụng 10 ký tự các số tự nhiên từ 0 đến 9, kết hợp với các dấu chấm, dấu phẩy để chỉ về lượng.

Trong dãy số thập phân: $d_{n-1} \dots d_2 d_1 d_0$ theo quy ước từ trái sang phải vị trí của chúng thể hiện hàng đơn vị, hàng chục, hàng trăm, hàng nghìn,... với phần nguyên và ngược lại là phần chục, phần trăm, phần nghìn,... với phần lẻ sau dấu phẩy.

Ví dụ: cho số 267,81 là số thập phân với phần nguyên là 267 và phần lẻ là 0,81 được biểu diễn như sau:

$$267,81_{(10)} = 2 \cdot 10^2 + 6 \cdot 10^1 + 7 \cdot 10^0 + 8 \cdot 10^{-1} + 1 \cdot 10^{-2}$$

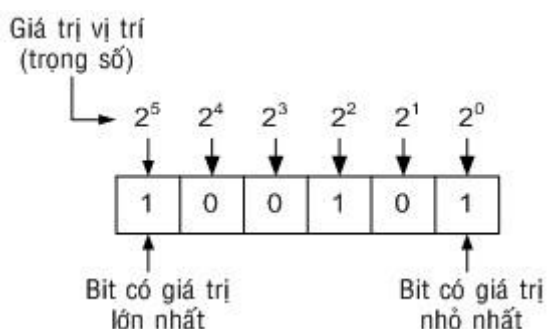


Trong một dãy thập phân có n số hạng thì trọng số của hai số hạng kề cận nhau chênh lệch 10 lần

* Hệ thống số nhị phân

Hệ thống số nhị phân sử dụng hai số tự nhiên 0 và 1 để diễn tả về lượng của một đại lượng nào đó. Một dãy số nhị phân được biểu diễn như sau: $b_{n-1} b_{n-2} \dots b_1 b_0$

Theo quy ước mỗi số hạng được gọi là 1 bit. Bit tận cùng bên trái gọi là MSB (bit có giá trị cao nhất), Bit tận cùng bên phải gọi là LSB (bit có giá trị thấp nhất).



Trong một dãy nhị phân có n số hạng, có 2^n giá trị khác nhau. Thì trọng số các bit từ thấp đến cao là 1, 2, 4, 8,... như vậy trọng số của hai số hạng kề cận nhau chênh lệch 2 lần.

Một nhóm của bit được gọi theo tên riêng sau:

Crum: 2 bit

Byte: 8 bit

Dynner: 32 bit

Nibble: 4 bit Deckte: 10 bit Nickle: 5

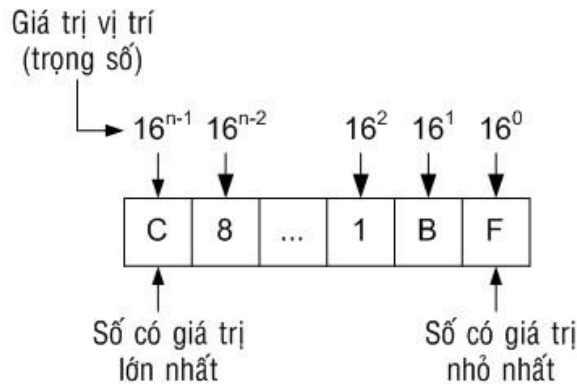
bit Playte: 16 bit * **Hệ thống thập**

lục phân (Hexa)

Hệ Hexa sử dụng 16 ký tự bao gồm 10 ký tự số tự nhiên và 6 ký tự in hoa đầu tiên: 0 1 2 3 4 5 6 7 8 9 A B C D E F để diễn tả 16 số thập phân từ 0 đến 15.

Lý do dùng hệ thập lục phân là vì một số nhị phân 4 bit có thể diễn tả được $2^4=16$ giá trị khác nhau.

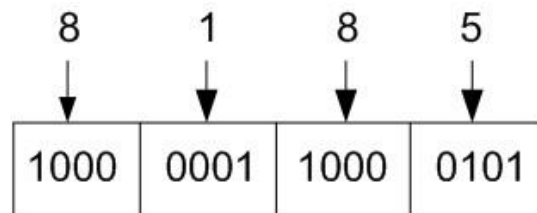
Một dãy số Hexa được biểu diễn như sau: $h_{n-1} h_{n-2} \dots h_1 h_0$



Trong một dãy Hexa có n số hạng, có 16^n giá trị khác nhau với giá trị thấp nhất là 0...000 và giá trị cao nhất là F...FFF. Thì trọng số các bit từ thấp đến cao là 1, 16, 256, 4096,... như vậy trọng số của hai số hạng kề cận nhau chênh lệch 16 lần.

*** Mã BCD:**

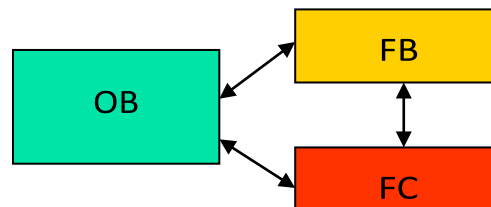
Nếu biểu diễn từng ký số của một số thập phân bằng giá trị nhị phân tương đương, kết quả là mã thập phân được mã hóa nhị phân (binary – code – decimal, viết tắt là BCD), vì ký số thập phân lớn nhất là 9, nên ta cần 4 bit để mã hóa mỗi ký số thập phân. Ví dụ:



b) Các vùng nhớ PLC S7-300

Bộ nhớ của S7-300 được chia thành 3 vùng chính :

- OB (Organisation Block): Miền chứa chương trình tổ chức.
- FB (Function Block): Miền chương trình con được tổ chức thành hàm và có khả năng trao đổi dữ liệu với bất cứ các khối khác.
- FC (Function): Miền chứa hàm, có biến hình thức để trao đổi với chương trình đã gọi nó.



Hình 1.15: Các vùng nhớ

c) Các miền dữ liệu:

✦ I (Process Input) :Miền bộ đệm các cổng vào số

- Giá trị ngõ vào từ các cảm biến được đọc và chứa vào I trước mỗi chu kỳ quét và thực hiện chương trình.

✚ **Q (Process Output) :Miền bộ đệm các cổng ra số**

- Kết thúc mỗi chu kỳ quét thực hiện chương trình, PLC chuyển giá trị logic điều khiển cơ cấu chấp hành ra miền nhớ Q .

✚ **M :Miền các biến cờ**

- Chương trình sử dụng miền nhớ này để lưu các tham số cần thiết.
- Có thể truy nhập theo bit, byte, word hay double word

✚ **T :Miền nhớ phục vụ bộ định thời**

- Lưu trữ giá trị thời gian đặt trước (PV-Preset Value). - Giá trị thời gian tức thời (CV – Current Value).
- Giá trị logic đầu ra của bộ Timer

✚ **C :Miền nhớ phục vụ đếm Counter** – Lưu trữ giá trị đặt

- trước (PV – Preset Value). - Giá trị đếm tức thời (CV – Current Value). - Giá trị logic đầu ra của bộ Counter

✚ **PI :Miền ngõ vào của các Module tương tự**

- Giá trị tương tự được Module đọc và lưu vào miền nhớ PI theo từng địa chỉ tương ứng. - Miền nhớ có thể truy nhập theo từng Byte (PIB), Word (PIW) hay Double Word (PID)

✚ **PQ :Miền ngõ ra của các Module tương tự**

- Các giá trị tương tự từ PLC xuất ra sẽ chuyển đến các vùng địa chỉ tương ứng của vùng nhớ này sau mỗi chu kỳ thực hiện chương trình.
- Miền nhớ có thể truy cập theo từng Byte (PQB), Word (PQW) hay Double Word (PQD)

*** Qui định các kiểu dữ liệu:**

Một chương trình trong S7-300 có thể sử dụng các kiểu dữ liệu sau:

- BOOL: với dung lượng là 1 bit và có giá trị là 0 hoặc 1 (đúng hoặc sai).

Đây là kiểu dữ liệu biến có hai giá trị.

- BYTE: gồm 8 bits, thường được dùng để biểu diễn một số nguyên dương trong khoảng từ 0 đến 255 hoặc mã ASCII của một ký tự.

Ví dụ: B#16#14 nghĩa là số nguyên 14 viết theo hệ đếm cơ số 16 có độ dài 1 byte.

- WORD: gồm 2 byte, để biểu diễn số nguyên dương từ 0 đến 65535 ($2^{16} - 1$).

- DWORD : Là từ kép có giá trị là : 0 đến $2^{32} - 1$.

- INT : cũng có dung lượng là 2 bytes, dùng để biểu diễn một số nguyên trong khoảng -32768 đến 32767 hay (2^{15} đến $2^{15} - 1$).

- DINT : gồm 4 bytes, dùng để biểu diễn số nguyên từ -2147483648 đến 2147483647 hay : (2^{31} đến $2^{31} - 1$).

- REAL : gồm 4 bytes, dùng để biểu diễn một số thực dấu phẩy động có giá trị là : $-3,4^{E38}$ đến $3,4^{E38}$.

Ví dụ: 1.234567e+13

- S5T (hay S5Time): khoảng thời gian, được tính theo giờ/phút/giây: ($2^{-31} - 1$ ms).

Ví dụ: S5t#2h_3m_0s_5ms.

Đây là lệnh tạo khoảng thời gian là 2 tiếng ba phút và 5 mili giây.

- TOD: Biểu diễn giá trị tức thời tính theo Giờ/phút/giây.

Ví dụ: TOD#5:30:00 là lệnh khai báo giá trị thời gian trong ngày là 5 giờ 30 phút.

- DATE: Biểu diễn thời gian tính theo năm / ngày / tháng.

Ví dụ: DATE#2003-6-12 Là lệnh khai báo

ngày 12 tháng 6 năm 2003.

- CHAR: biểu diễn một hoặc nhiều ký tự (nhiều nhất là 4 ký tự) (ASCII – code).

Ví dụ: ABCD

*** Tóm lại một số kiểu dữ liệu thông dụng là: -**

Boolean

- Byte, Word (2Bytes), Double Words (4 Bytes).
- S5 Timer.
- Counter.
- Time, Date, Time of Date
- Integer, Double Integer

Dạng	Kích thước	Dạng Format	Tầm và ký hiệu (từ giá trị nhỏ nhất đến giá trị lớn nhất)	Ví dụ
BOOL (bit)	1	Boolean Text	True/Fasle	True
BYTE (Byte)	8	Thập lục phân	B#16#0 đến B#16#FF	B#16#10 Byte#16#10
Word (Word)	16	Nhị phân	2#0 đến 2#1111_1111_1111_1111	2#0001_0010_0000_0011
		Thập lục phân	W#16#0 đến W#16#FFFF	W#16#1CBF Word W#16#1CBF
		BCD	C#0 đến C#999	C#998
		Thập phân	B#(0,0) đến B#(255,255)	B#(10,20) Byte#(10,20)

Dword	32	Nhi phân Thập lục phân Thập phân (không dấu)	2#0 2#1111_1111_1111_1111_1111_1111_1111_1111 W#16#0000_0000 W#16#FFFF_FFFF B#(0,0,0,0) B#(255,255,255,255)	đến đến đến	2#0010_0111_1001_0000_0011_0100_1111_1000 DW#16#00A2_0FAB DWord W#16#1CBF B#(1,14,65,245) Byte#(1,14,65,245)
INT Interger	16	Thập phân có dấu	-32768 đến 32767		2
DINT Double Interger	32	Thập phân có dấu	L#-2147483648 đến L#-2147483647		L#2
CHAR character	8	Ký tự	'A','B','c',...		'e'
S5TIMER Simatic Timer	16	S5Timer với đơn vị là 10ms	S5T#0H_0M_0S_10MS đến S5T#2H_46M_30S_0MS		S5T#1M S5TIME#1M
TIME (EIC time)	32	EIC Time với đơn vị là 1ms (số)	T#24D_20H_31M_23S_468MS đến		T#1H_1M TIME#1H_1M
		interger có dấu)	T#24D_20H_31M_23S_467MS		
DATE (EIC date)	16	Ngày hệ EIC với đơn vị là 1 ngày)	D#1990-1-1 đến D#2168-12-31		D#1994-2-15 DATE#1994-2-15
TIME_OF_DAY (Time of day)	32	Thời gian trong một ngày với đơn vị là 1ms	TOD#0:0:0.0 đến TOD#23:59:59.999		TOD#1:10:3.6 TIME_OF_DAY#1:10:3.6

d) Cách truy cập địa chỉ miền nhớ: Địa chỉ ô nhớ trong Step7-300 gồm hai phần: phần chữ và phần số. Ví dụ:



✚ **Phần chữ:** chỉ vị trí (tên miền nhớ) và kích thước ô nhớ.

- **Ô nhớ M:**

+ M: Ô nhớ nội có kích thước 1 Bit.

+ MB: Ô nhớ nội có kích thước 1 Byte (8Bits).

+ MW: Ô nhớ nội có kích thước 2Bytes (16 Bits).

+ MD: Ô nhớ nội có kích thước 4 Bytes (32Bits)

- **Ô nhớ I:**

+ I: Ô nhớ ngõ vào số có kích thước 1 Bit.

+ PIB: Ô nhớ ngõ vào có kích thước 1 Byte (8Bits).

+ PIW: Ô nhớ ngõ vào có kích thước 2Bytes (16 Bits).

+ PID: Ô nhớ ngõ vào có kích thước 4 Bytes (32Bits)

- **Ô nhớ Q:**

+ Q: Ô nhớ ngõ ra số có kích thước 1 Bit.

+ PQB: Ô nhớ ngõ ra số có kích thước 1 Byte (8Bits).

+ PQW: Ô nhớ ngõ ra số có kích thước 2Bytes (16 Bits).

+ PQD: Ô nhớ ngõ ra số có kích thước 4 Bytes (32Bits)

✚ **Phần số:** chỉ địa chỉ của Byte hoặc Bit trong miền nhớ đã được xác định.

- Nếu ở phần chữ đã xác định là ô nhớ truy cập theo Bit thì ở phần số sẽ gồm hai phần cách nhau bằng dấu chấm:

+ Địa chỉ của Byte.

+ Số thứ tự của Bit trong Byte đó.

✚ **Ví dụ:**

- I1.3 : Địa chỉ Bit thứ 4 trong Byte thứ 2 của vùng ô nhớ bộ đệm **ngõ vào số**

- M101.5 : Địa chỉ Bit thứ 6 trong Byte thứ 102 của vùng **ô nhớ nội**.

- Q4.5 : Địa chỉ Bit thứ 6 trong Byte thứ 5 của vùng nhớ bộ đệm **ngõ ra số**.

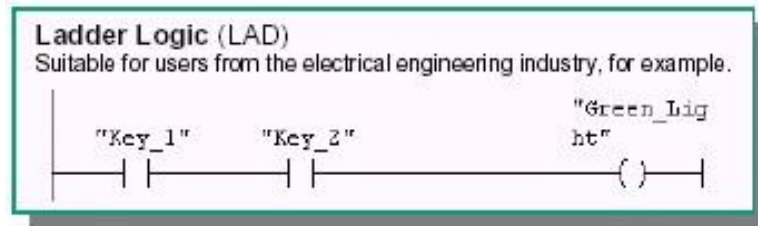
- PIW120: Địa chỉ 2 Byte thứ 120 (byte 120 và 121) trong vùng nhớ bộ đệm **ngõ vào tương tự**.

- PQD10: Địa chỉ 4 Byte thứ 10 (byte 10, 11, 12, 13) trong vùng nhớ bộ đệm **ngõ ra tương tự**

4. Ngôn ngữ lập trình cho PLC

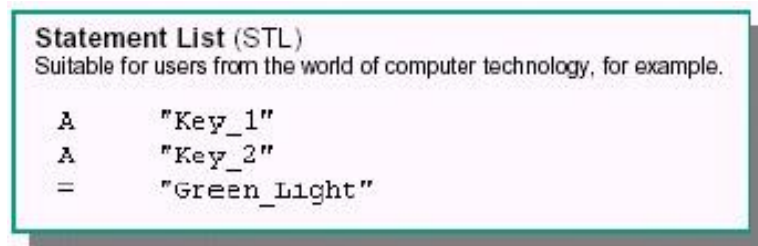
Tùy theo từng loại PLC của các hãng sx khác nhau mà có ngôn ngữ lập trình khác nhau. PLC Siemens S7-300 có 3 loại ngôn ngữ lập trình cơ bản sau:

- Ngôn ngữ “hình thang”, ký hiệu là LAD (Ladder logic): Đây là ngôn ngữ đồ họa thích hợp với những người quen thiết kế mạch logic.



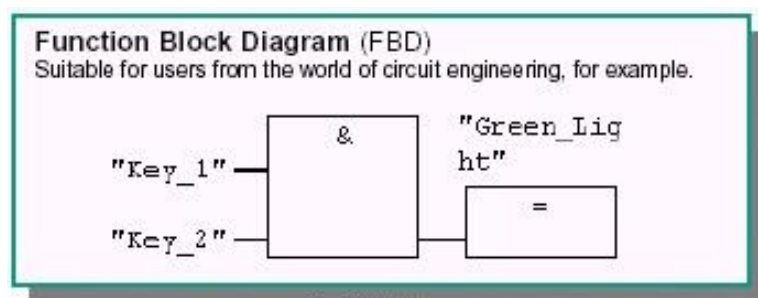
Hình 1.16: Ngôn ngữ lập trình LAD

- Ngôn ngữ “liệt kê lệnh”, ký hiệu là STL (Statement list): Đây là dạng ngôn ngữ lập trình thông thường của máy tính. Một chương trình được ghép gởi nhiều câu lệnh theo một thuật toán nhất định, mỗi lệnh chiếm một hàng và đều có cấu trúc chung là “tên lệnh” + “toán hạng”.



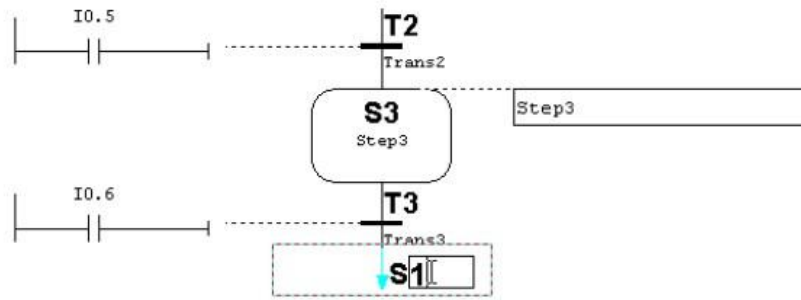
Hình 1.17: Ngôn ngữ lập trình STL

- Ngôn ngữ “hình khối”, ký hiệu là FBD (Function Block Diagram): Đây cũng là ngôn ngữ đồ họa thích hợp với những người quen thiết kế mạch điều khiển số.



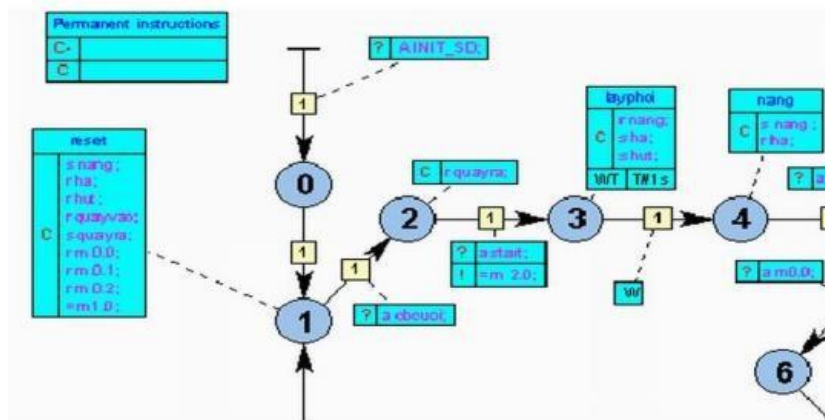
Hình 1.18: Ngôn ngữ lập trình FBD

- Ngôn ngữ GRAPH: Đây là ngôn ngữ lập trình cấp cao dạng đồ họa. Cấu trúc chương trình rõ ràng, chương trình ngắn gọn. Thích hợp cho người trong ngành cơ khí vốn quen với giản đồ Grafcet của khí nén.



Hình 1.19: Ngôn ngữ lập trình GRAPH

- Ngôn ngữ High GRAPH: Là dạng ngôn ngữ lập trình phát triển từ ngôn ngữ lập trình GRAPH.



Hình 1.20: Ngôn ngữ lập trình High GRAPH

Bài 2: CÀI ĐẶT VÀ SỬ DỤNG PHẦN MỀM S7-300

Mục tiêu của bài:

- Trình bày được các bước cài đặt và sử dụng phần mềm S7 – 300.
- Cài đặt và sử dụng thành thạo phần mềm S7 – 300 để lập trình và mô phỏng.
 - Thao tác cài đặt và lập trình chính xác, chương trình ngắn gọn, dễ hiểu.

Nội dung của bài:

I. Cài đặt phần mềm Step7

A. PHẦN LÝ THUYẾT:

1. Giới thiệu phần mềm Step7

Step7 là một phần mềm hỗ trợ:

- Khai báo cấu hình phần cứng cho một trạm PLC thuộc họ Simatic S7300/400.
- Xây dựng cấu hình mạng gồm nhiều trạm PLC S7-300/400 cũng như thủ tục truyền thông giữa chúng.
- Soạn thảo và cài đặt chương trình điều khiển cho một hoặc nhiều trạm.
- Quan sát việc thực hiện chương trình điều khiển trong một trạm PLC và gỡ rối chương trình.

Ngoài ra Step7 còn có cả một thư viện đầy đủ với các hàm chuẩn hữu ích, phần trợ giúp online rất mạnh có khả năng trả lời mọi câu hỏi của người sử dụng về cách sử dụng Step7, về cú pháp lệnh trong lập trình, về xây dựng cấu hình cứng của một trạm cũng như của một mạng gồm nhiều trạm PLC.

Những chú ý cài đặt chứa thông tin quan trọng mà bạn cần trong quá trình cài đặt STEP 7 V5.4. Cần đọc những chú ý này trước khi cài đặt phần mềm.

Yêu cầu.

Trong quá trình làm việc với phần mềm STEP7 bạn cần:

Hệ điều hành	Yêu cầu tối thiểu		
	Processor	RAM	Graphics
MS Windows 2000 Professional	600MHz	512 MB *	XGA 1024x768 16 Bit color depth
MS Windows XP Professional	600MHz	512 MB *)	XGA 1024x768 16 Bit color depth
MS Windows Server 2003	2.4 GHz	1 GB	XGA 1024x768 16 Bit color depth

MS Windows Vista Business	1 GHz	1GB **)	XGA 1024x768 16 Bit color depth
MS Windows Vista Ultimate	1 GHz	1GB **)	XGA 1024x768 16 Bit color depth

2. Các bước cài đặt phần mềm Step7

Bước 1: Đưa đĩa CD STEP 7 vào ổ đĩa.

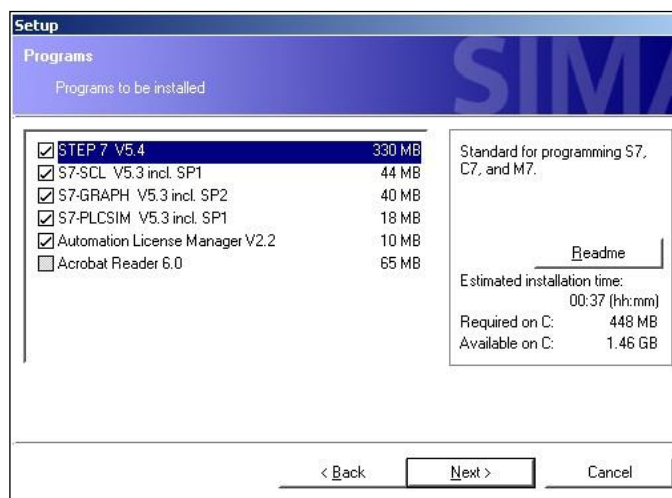
Bước 2: Nhấn đúp chuột vào tệp SETUP.EXE trên đĩa CD STEP 7 từ Windows Explorer, xuất hiện hộp thoại:



Hình 2.1a: Hộp thoại setup chọn ngôn ngữ

Chọn ngôn ngữ tiếng Anh (English), click **Next**. Tiếp tục click Next cho đến khi xuất hiện hộp thoại:

Bước 3: Chọn các chương trình cần cài đặt




Hình 2.1b: Hộp thoại setup chọn các chương trình cần cài đặt

Tiếp tục click nút **Next** và chờ cho đến khi cài đặt xong.

Bước 4: Copy thư mục **AX NF ZZ** vào địa chỉ cài đặt (ví dụ: C:\Program Files\Siemens\Step 7)

Sau khi cài đặt xong trên màn hình desktop xuất hiện biểu tượng của phần mềm.



Đồng thời trong  menu cũng xuất hiện thư mục Simatic với tất cả thành phần từ cài đặt cấu hình, mô phỏng, chuyển đổi dữ liệu,...

*** Các lưu ý khi sử dụng phần mềm:**

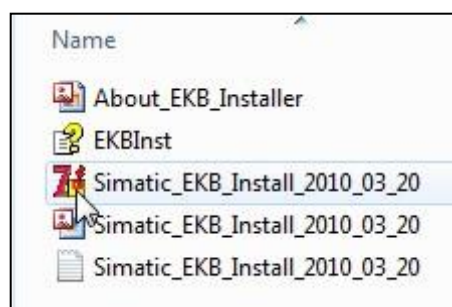
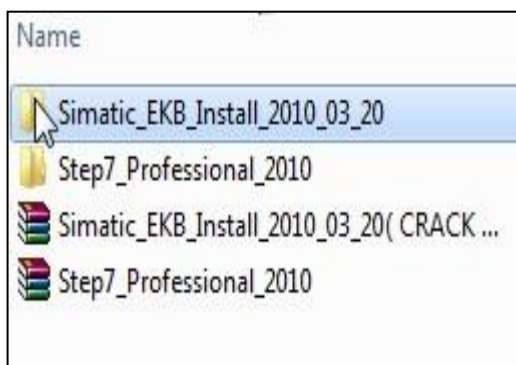
- Sau khi cài đặt xong, muốn sử dụng STEP 7 V5.4, Windows phải được khởi động lại. Chỉ sau khi khởi động lại Microsoft Windows thì các ghi nhận về phần mềm mới được kích hoạt. Nếu ta không khởi động lại Windows, STEP 7 V5.4 không thể chạy chính xác và dữ liệu có thể bị mất. Nếu quá trình cài đặt bị bỏ dở (abort), ta cũng cần phải khởi động lại Windows.

- STEP 7 tự đăng ký (ghi) bản thân nó vào trong các file hệ thống của Windows. Ta không thể chuyển hoặc thay đổi tên các file và thư mục của STEP 7 bằng cách sử dụng các tiện ích của Microsoft Windows như Explorer hoặc sửa đổi dữ liệu của STEP 7 đã đăng ký trong Windows. Chương trình có thể sẽ chạy không chính xác nữa sau khi bị sửa đổi.

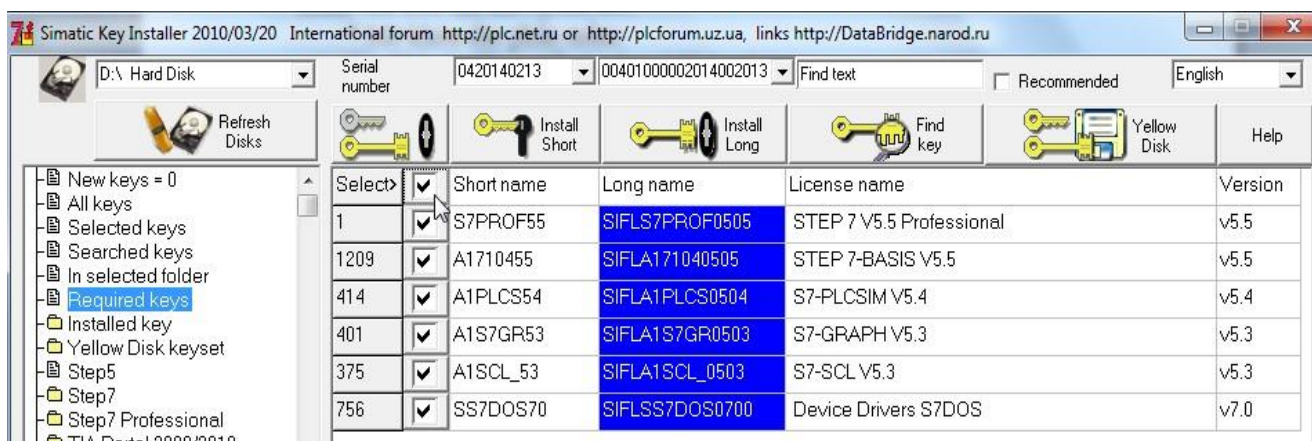
*** Mở khóa cho phần mềm:**

Sau khi cài đặt, ta tiến hành mở khóa như sau:


- Mở file S7-300, chọn thư mục Simatic_EKB_Instal_(2010_03_20)/file Simatic_EKB_Instal_(2010_03_20)

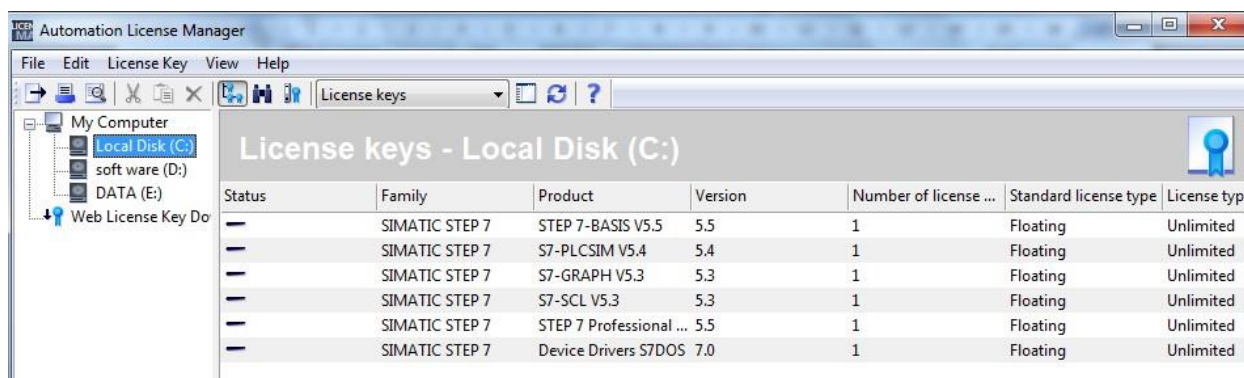


Hình 2.2a: Chọn thư mục Simatic_EKB_Instal
Cửa sổ Simatic_EKB_Installer hiện ra, chọn Select all/install long/all



Hình 2.2b: Tiến hành mở khóa

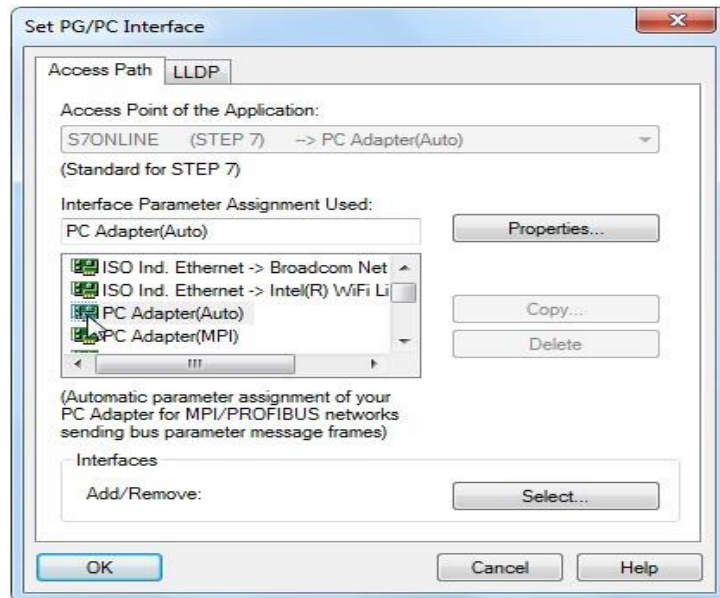
- Trở ra màn hình desktop, chọn  để vào cửa sổ Automation license manager để xem kết quả, nếu tất cả các mục đều hiện “Unlimited” như hình sau thì việc mở khóa thành công:



Hình 2.2c: Cửa sổ Automation License Manager sau khi mở khóa thành công

*** Khai báo PC Ad,aptor:**

Trong cửa sổ Simatic manager, chọn Option/Set PG/PC Interface..., hiện lên cửa sổ Set PG/PC Interface, chọn PC Adaptor(Auto)/nhập OK.



Hình 2.3: Hộp thoại khai báo adaptor

Nếu không có PC Adaptor(Auto) thì nhấp chọn Select để vào cửa sổ Install/Remove interface để tìm và cài đặt PC Adaptor.

B. PHẦN THỰC HÀNH:

* Bài tập thực hành:

Hãy thực hiện cài đặt phần mềm Step 7 theo các bước hướng dẫn trên.

Các bước thực hiện:



- **Bước 1:** Đưa đĩa CD STEP 7 vào ổ đĩa.
- **Bước 2:** Nhấn đúp chuột vào tệp SETUP.EXE để cài đặt
- **Bước 3:** Chọn các chương trình cần cài đặt
- **Bước 4:** Copy thư mục **AX NF ZZ** vào địa chỉ cài đặt (ví dụ: C:\Program Files\Siemens\Step 7)

II. Sử dụng phần mềm Step7:

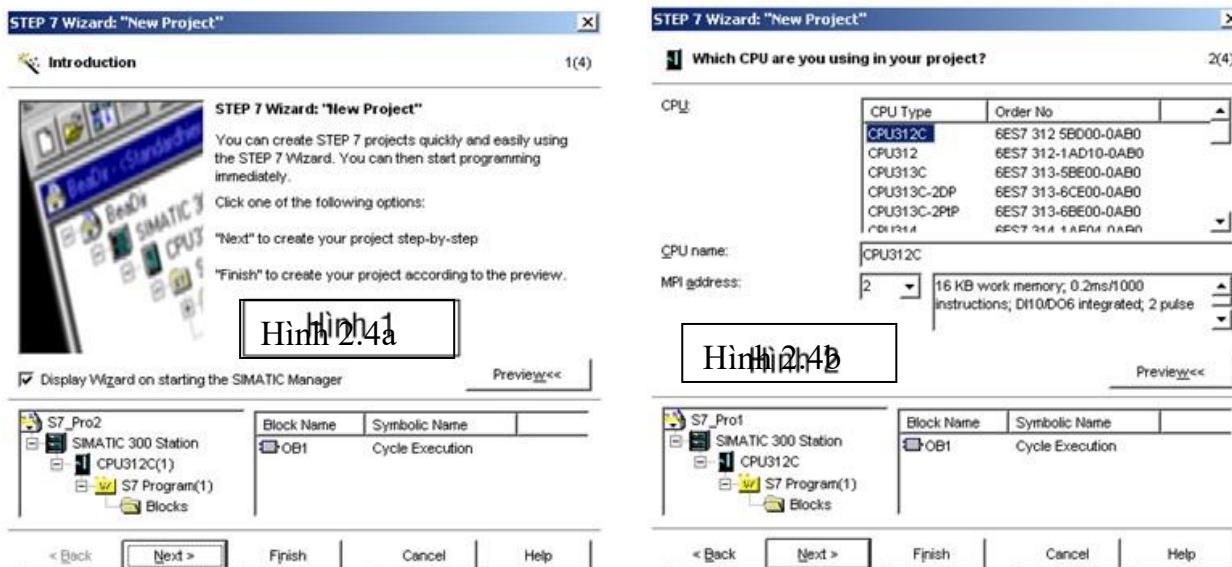
A. PHẦN LÝ THUYẾT:

1. Khởi động chương trình tạo project:

1.1. Cách 1: Lưu Project mặc định trong ổ C:/Program Files/...

Để khởi động chương trình, ta có thể thực hiện theo các cách  vào **Start/Simatic/Simatic Manager** hoặc nhấp đôi vào biểu tượng Step 7  trên màn hình, xuất hiện cửa sổ **Simatic Manager** cùng với hộp thoại “New Project” (hình 2.4a). Nếu không thấy hộp thoại này thì vào **menu File/chọn New Project Wizard**. Để hộp thoại “New Project” luôn xuất hiện khi khởi động chương trình, ta đánh dấu nhắc chọn chế độ “Display Wizard on starting the SIMATIC Manager” (hiện hộp thoại Wizard khi khởi động SIMATIC Manager). Tại đây nếu ta nhấp **Finish** thì việc

khởi tạo chương trình sẽ được **mặc định** theo preview bên dưới (tên project, ngôn ngữ lập trình... đều mặc định). Nếu nhấp Next thì khởi tạo Project từng bước. Ở đây ta chọn cách tạo Project từng bước nên nhấp Next.

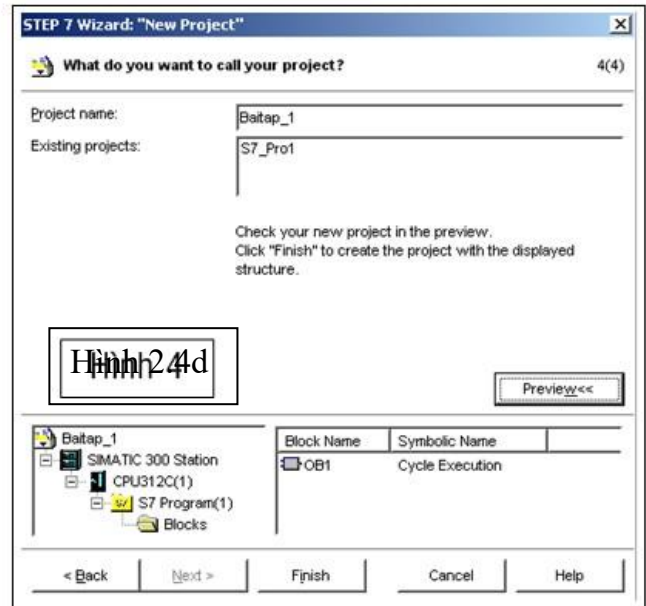
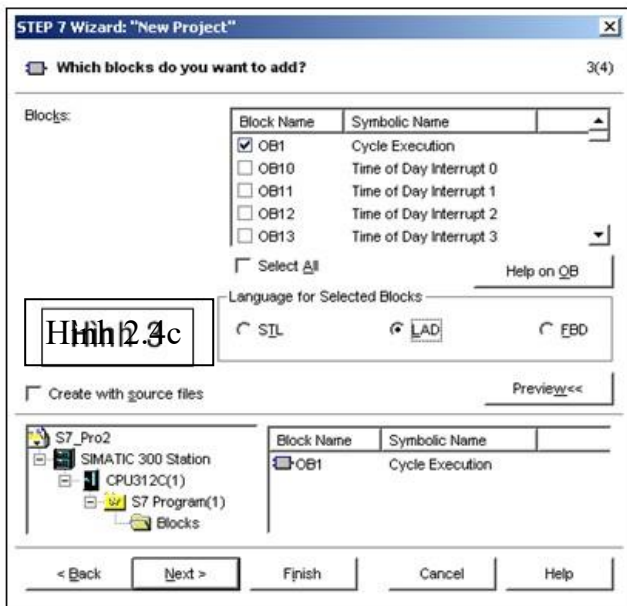


Hình 2.4a,b: Giới thiệu về STEP 7 Wizard và chọn CPU

Để khởi tạo 1 project mà project này thực thi được trên phần cứng sau khi lập trình, ta phải thực hiện các bước lựa chọn trên phần mềm sao cho tương thích với phần cứng hiện có, ở đây ta phải chọn CPU và khối lập trình.

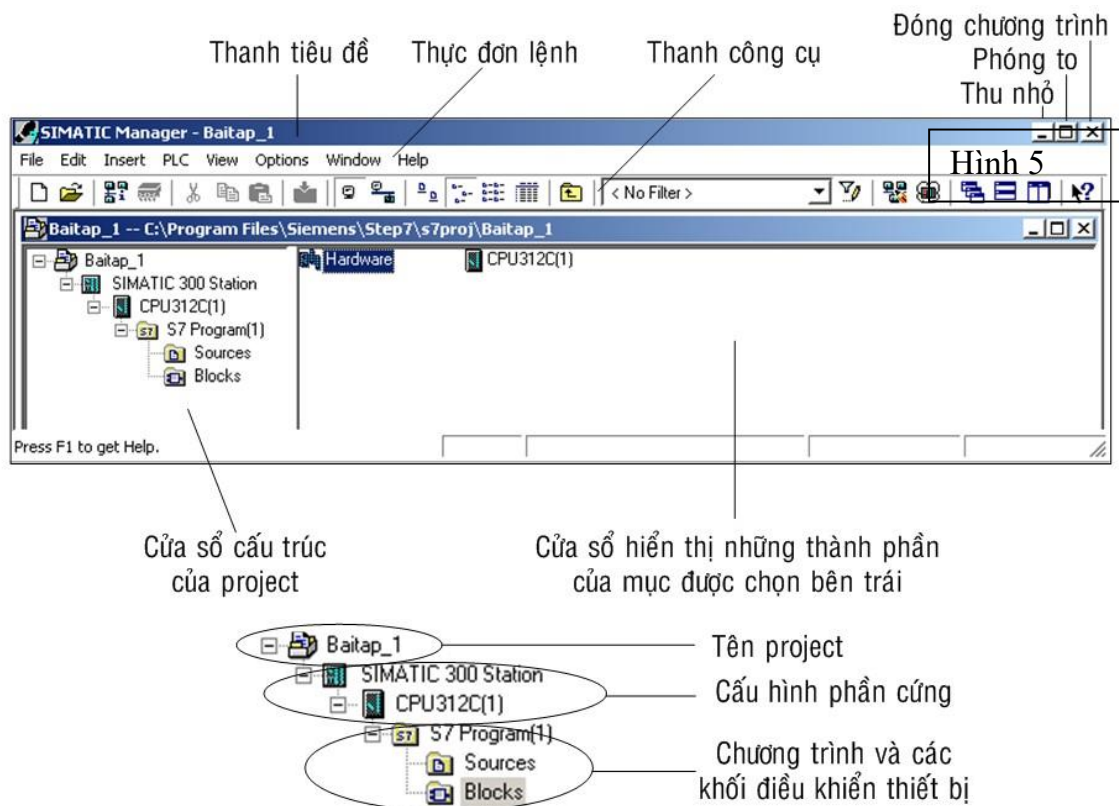
- CPU hiện có tại xưởng là loại CPU312C (hình 2.4b) nên ta chọn loại này, sau đó click **Next**.

- Bước tiếp theo là chọn khối cần lập trình (Ví dụ khối OB1) và chọn ngôn ngữ lập trình (ví dụ ngôn ngữ LAD) (hình 2.4c) sau đó click **Next**.



Hình 2.4c,d: Chọn ngôn ngữ lập trình và đặt tên cho project

- Đặt tên cho chương trình trong project name (hình 2.4d) sau đó click **Finish**. Chương trình trở lại cửa sổ SIMATIC Manager như hình 2.5. Ta có thể bỏ qua bước đặt tên, chương trình sẽ cho tên mặc định là S7_Pro và sau đó là số thứ tự của project được tạo ra, tuy nhiên để dễ quản lý chương trình thì ta không nên bỏ qua bước này.



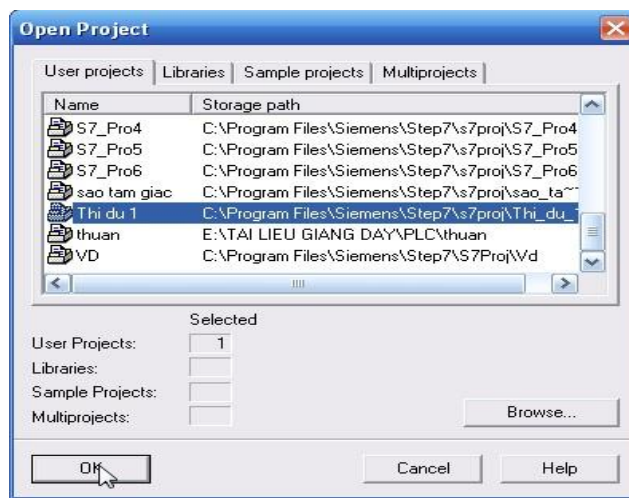
Hình 2.5: Cấu trúc một Project

* **Cấu trúc một Project của Simatic**

Cấu trúc cơ bản của 1 project được thể hiện như hình trên, trước tiên ta thấy tên project là “Baitap_1”, đây là tên do người lập trình đặt. Sau đó là cấu hình phần cứng của trạm SIMATIC 300 có CPU312C do người sử dụng chọn và phần bên dưới thể hiện chi tiết hơn chương trình và các khối hiển thị.

*** Mở một project đã tồn tại sẵn:**

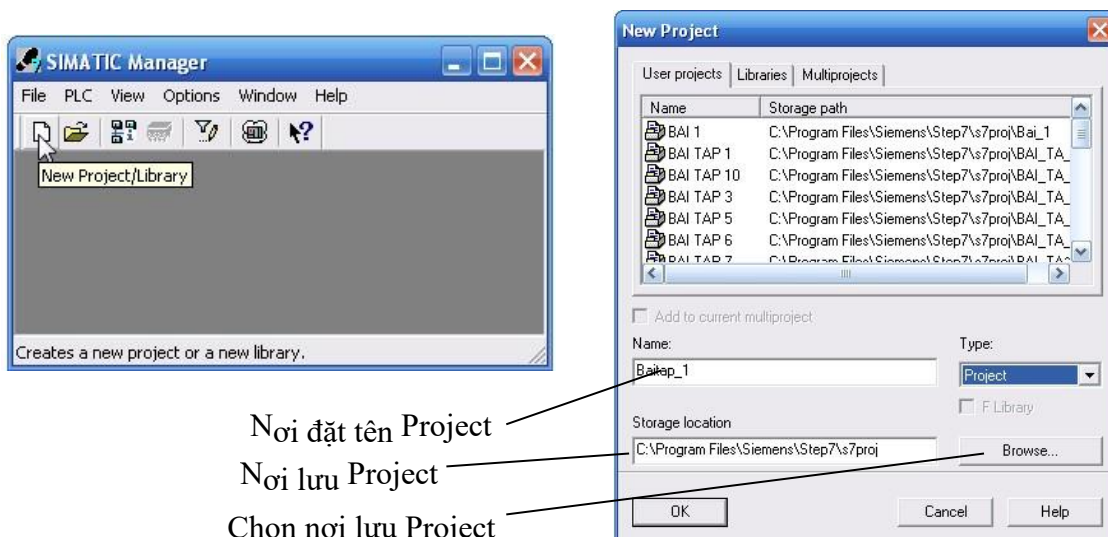
Đối với trường hợp mở 1 project đã tồn tại sẵn, ta bỏ qua tất cả các thao tác chọn cấu hình và đặt tên và thực hiện như sau: sau khi vào **Start/Simatic/Simatic Manager** hoặc nhấp đôi vào biểu tượng Step 7 trên màn hình, xuất hiện cửa sổ **Simatic Manager** cùng với hộp thoại “New Project”. Ta đóng cửa sổ “New project” lại để vào Menu File/chọn open hoặc nhấp vào biểu tượng Open Project/Library, lập tức xuất hiện hộp thoại Open Project. Trong hộp thoại Open Project, chọn tên file cần mở, nhấp OK.



Hình 2.6: Cửa sổ open project

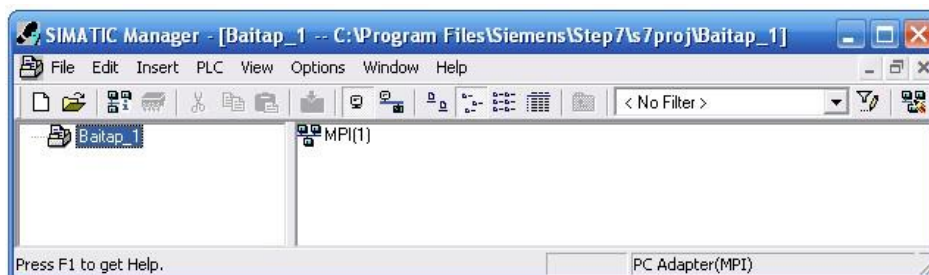
1.2. Cách 2: Lưu Project vào các ổ đĩa khác:

Ta có thể vào Menu File/chọn New hoặc nhấp vào biểu tượng mở project mới “New Project/Library” sẽ xuất hiện hộp thoại New Project:



Hình 2.7a: Mở project mới trong cửa sổ chính

Tại đây ta có thể đặt tên cho Project, chọn nơi lưu Project theo ý muốn, nhấp OK. Màn hình trở lại cửa sổ chính như sau:

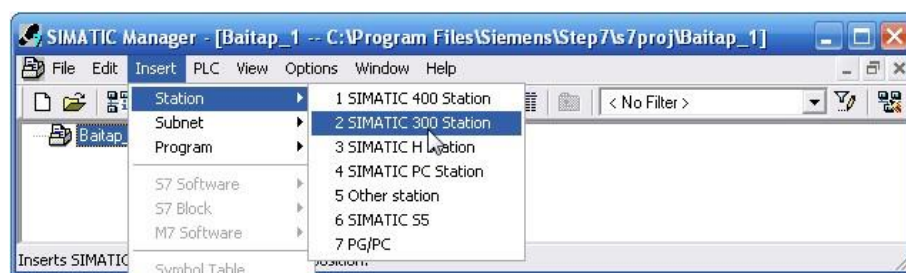


Hình 2.7b: Project mới sau khi được mở là project rỗng.

Sau khi khai báo xong một Project mới, trên màn hình sẽ xuất hiện Project đó nhưng ở dạng rỗng (chưa có gì trong Project), điều này ta nhận biết được qua biểu tượng thư mục bên cạnh tên Project giống như một thư mục rỗng của Window.

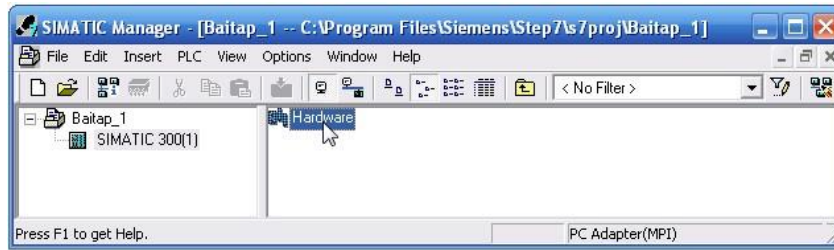
Do là project rỗng nên ta phải xây dựng cấu hình phần cứng cho một trạm PLC. Điều này là không bắt buộc, ta có thể không cần khai báo cấu hình cứng cho trạm mà đi ngay vào phần chương trình ứng dụng. Song công việc này nên làm vì khi có cấu hình trong Project, lúc bật nguồn PLC, hệ điều hành của S7-300 bao giờ cũng đi kiểm tra các module hiện có trong trạm, so sánh với cấu hình mà ta xây dựng và nếu phát hiện thấy sự không đồng nhất sẽ phát ngay tín hiệu báo ngắt lỗi chứ không cần phải đợi tới khi thực hiện chương trình ứng dụng.

Vào Insert/ chọn Station/ chọn SIMATIC 300 Station. Có thể làm việc này bằng cách nhấp phải vào biểu tượng có tên Project/chọn Insert New Object/ chọn SIMATIC 300 Station:



Hình 2.8: Chọn cấu hình phần cứng cho một trạm PLC.

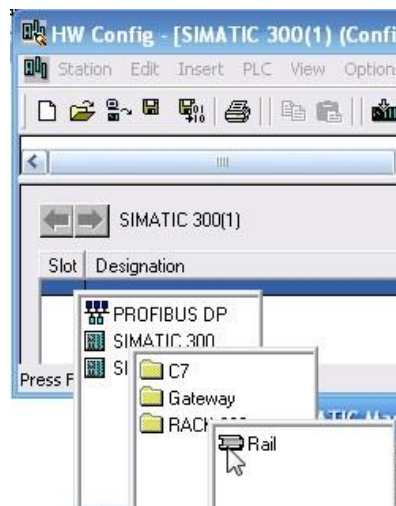
* **Khai báo cấu hình phần cứng:** -
Nhấp chuột tại biểu tượng Hardware.



Hình 2.9: Vào cửa sổ HW Config.

Xuất hiện cửa sổ HW Config, tuy nhiên khác với các bước trên, cửa sổ này không có sẵn module CPU và cũng không có bất cứ Slot nào, để đưa các module và slot, ta phải tạo các slot:

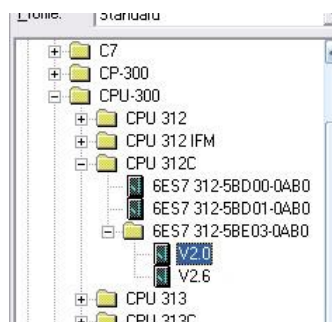
- Chọn Insert/Object hoặc nhấp phải vào Slot duy nhất trên màn hình/chọn SIMATIC 300/Rack 300/rail.



Hình 2.10: Tạo các slot trong HW Config.

Trên màn hình sẽ xuất hiện 11 Slot, bỏ Slot 1 (vì không dùng module PS), nhấp chuột chọn Slot 2, sau đó chọn module CPU theo cấu hình hiện có tại xưởng (CPU 312C, V2.0 hoặc V2.6)

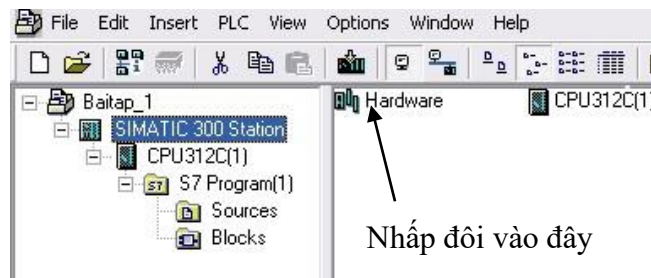
- Các module tiếp theo được chọn theo hướng dẫn mục 2 (thiết lập phần cứng)



Hình 2.11: Chọn version cho CPU

2. Thiết lập phần cứng:

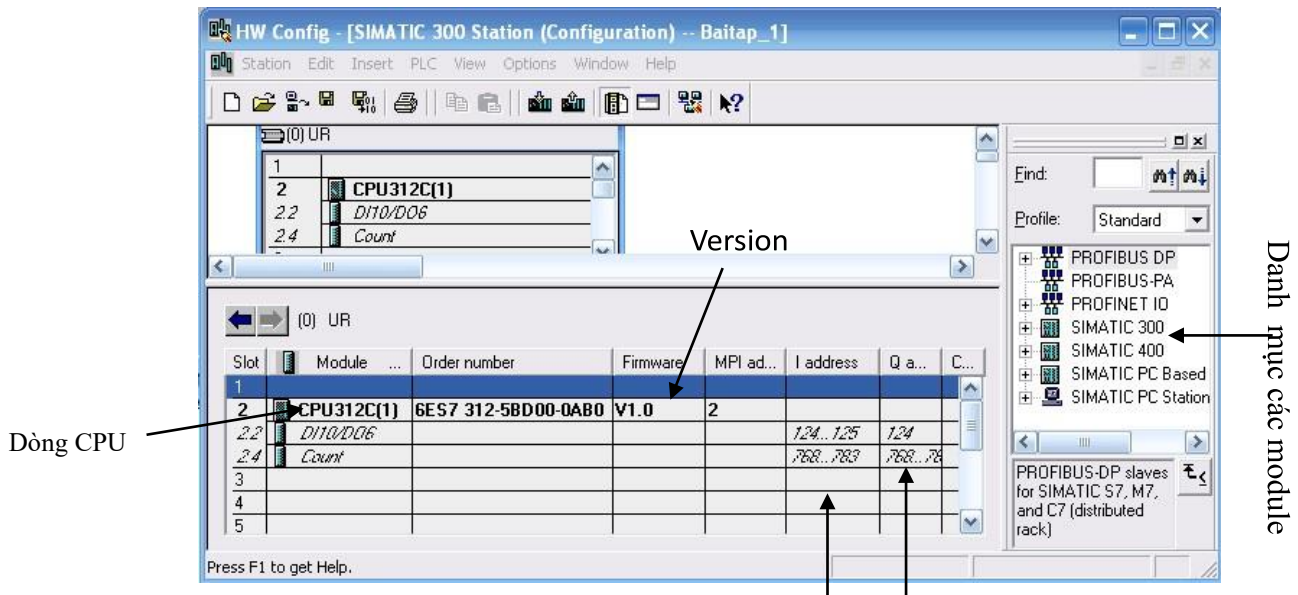
Trong quá trình khởi tạo project, ta chỉ chọn cấu hình CPU mà chưa chọn version và các modul mở rộng nên ở đây ta phải thiết lập phần cứng cho các phần còn lại: trở lại cửa sổ SIMATIC Manager, chọn SIMATIC 300 (bên trái cửa sổ)/ nhấp đôi vào biểu tượng Hardware bên phải cửa sổ (xem hình) sẽ xuất hiện cửa sổ HW Config.



Hình 2.12: Vào Hardware để thiết lập phần cứng

* Trước tiên, ta thiết lập phần cứng cho CPU:

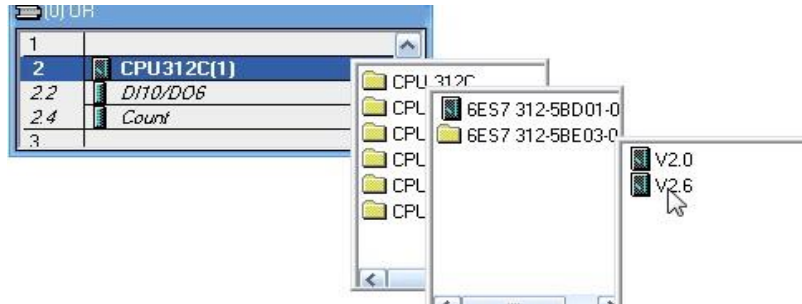
- Trong cửa sổ HW Config, đã có dòng CPU312C (do đã được chọn trong khi khởi tạo Project). Để chương trình có thể hoạt động được khi download về PLC thì phải chọn lại version cho phù hợp và để dễ dàng quản lý chương trình khi lập trình, ta nên thay đổi địa chỉ vào/ra chứ không nên để mặc định.



Địa chỉ vào/ra

Hình 2.13: Cửa sổ HW Config, nơi thiết lập phần cứng.

- Chọn version: nhấp chọn CPU312C (dòng 2)/nhấp phải/chọn Replace object/chọn CPU312C/ chọn 6ES7 312-5BE03-OABO(thư mục màu vàng) /chọn V2.0 hoặc V2.6/yes.



Hình 2.14: Chọn version cho CPU.

- Thay đổi địa chỉ vào/ra: Nhấp đôi vào DI10/DO6 (dòng 2.2)/hiện cửa sổ Properties/Chọn addresses/bỏ dấu nhắc mặc định (system default) của địa chỉ vào và ra/cài đặt địa chỉ mong muốn (thường bắt đầu từ byte 0) * **Thiết lập phần cứng cho các module mở rộng:**

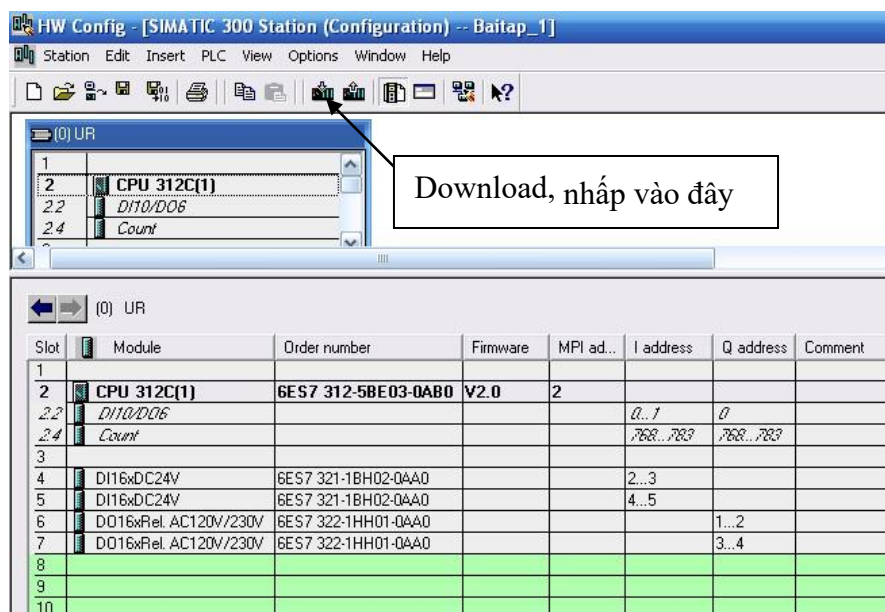
- Module mở rộng ngõ vào (module thứ 4): nhấp chuột chọn dòng thứ 4/bên cửa sổ phải, trong danh mục các module, mở SIMATIC 300/ mở SM 300/ mở DI 300/chọn module SM 321 DI 16 x DC 24V / 321 – 1BH02-0AA0 (theo PLC hiện có)/nhấn phím enter.

- Module thứ 5: tương tự như trên.

- Module mở rộng ngõ ra (module thứ 6): nhấp chuột chọn dòng thứ 6/bên cửa sổ phải, trong danh mục các module, mở SIMATIC 300/ mở SM 300/ mở DO 300/chọn module SM 322 DO 16 x RELAY AC 230V / 322-1HH01-0AA0 (theo PLC hiện có)/nhấn phím enter.

- Module thứ 7: tương tự như trên.

Sau khi hoàn thành, cửa sổ có hình ảnh như sau:



Hình 2.15: Cửa sổ HW Config sau khi đã thiết lập phần cứng và cài đặt địa chỉ

Lưu ý:

- Việc thiết lập phần cứng có thể không cần nếu ta chỉ viết chương trình và cho chạy mô phỏng. Tuy nhiên việc này là bắt buộc phải thực hiện trước khi download chương trình lên PLC cho hệ thống hoạt động, vì đây được xem là thao tác “bắt tay” giữa phần cứng và phần mềm để hai phần này có sự đồng bộ thì PLC mới hoạt động được.

- Module PS (thứ 1) và IM (thứ 3) không có trên mô hình nên không cần chọn khi thiết lập phần cứng.

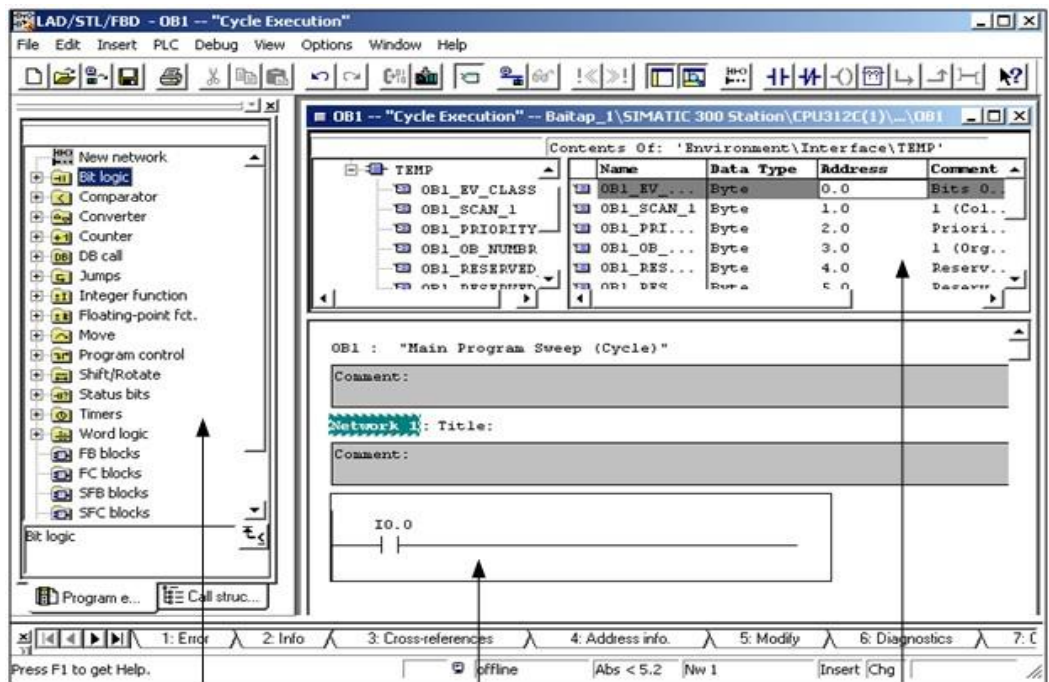
- Khi một byte được cấp cho một vùng nhớ, nếu ta dùng không hết 8 bit của nó thì sang vùng nhớ khác cũng phải khai báo byte khác mà không được sử dụng các bit thừa của byte trước. Thí dụ byte 0 và byte 1 có tổng cộng 16 bit, được cấp cho 10 ngõ vào của CPU, như vậy sau khi gán địa chỉ cho 10 ngõ vào, ta thừa 6 bit địa chỉ (từ bit thứ 2 đến bit thứ 7 của byte 1) nhưng khi khai báo địa chỉ cho các module mở rộng ngõ vào thì ta không thể sử dụng các bit còn lại này của byte 1 mà phải cấp cho nó vùng nhớ của byte 2,3...

3. Viết chương trình điều khiển:

Ở cửa sổ Simatic Manager ta chọn Block ở cửa sổ bên trái, sau đó click đôi vào biểu tượng khối OB1 ở cửa sổ phải sẽ hiện lên cửa sổ lập trình (LAD/LST/FBD).



Hình 2.16a: chọn OB1 trong cửa sổ Simatic Manager để lập trình



Bảng các công cụ lập trình Cửa sổ soạn thảo Bảng khai báo biến và tham số khối

Hình 2.16b: Cửa sổ lập trình OB1

- Bảng khai báo biến và tham số khối: dùng để khai báo biến và tham số cho khối lập trình

- Bảng các công cụ lập trình: chứa các công cụ cần thiết cho lập trình như: các lệnh logic điểm, Timer, Counter, các lệnh toán học (số nguyên, số thực), các lệnh so sánh, các lệnh chuyển đổi,...

- Cửa sổ soạn thảo: dùng để chứa chương trình điều khiển. Nó gồm nhiều Network, mỗi Network là một trạng thái của chương trình điều khiển.

Để lập trình phần tử nào thì ta click đôi vào phần tử đó trên bảng công cụ lập trình, sau đó nhập địa chỉ cho phần tử đó. **Chú ý:**


- Mỗi byte địa chỉ có 8 bit, nếu dùng hết byte này thì phải sang byte khác.


- Có thể lấy các lệnh thông dụng bằng cách nhấp vào biểu tượng trên thanh công cụ.

- Chuyển ngôn ngữ lập trình bằng cách vào menu View/LAD hoặc LST hoặc FBD.

4. Mô phỏng chương trình

- Để xem chương trình hoạt động có chính xác không, ta mở chương trình mô phỏng để xem trước các hoạt động của nó. Cửa sổ mô phỏng được gọi từ cửa sổ Simatic Manager hoặc có thể mở bằng cách vào Start/Simatic/Step 7/S7 PLCSIM Simulating.

- Để mô phỏng chương trình ta phải download chương trình về PLCSIM, trên cửa sổ lập trình ta click nút  (hoặc menu **PLC/Download**).

- Để kích hoạt chức năng kiểm tra và quan sát: Click vào biểu tượng mắt kính  trên thanh công cụ hoặc vào menu **Debug -> Monitor**.

- Chạy chương trình mô phỏng: trên bảng CPU ta click nút **RUN**.

- Để nhập tín hiệu vào mô phỏng ta click vào nút **địa chỉ vào** trên bảng IB (ví dụ địa chỉ ta nhập là byte 0, bit 0 và 1, lúc này các lệnh vào trong phần lập trình có địa chỉ vào là I0.0 và I0.1).

Nếu địa chỉ ngõ ra là byte 0, bit 0 thì trên hộp thoại mô phỏng ngõ ra Q0.0 sẽ xuất tín hiệu ra.

Trạng thái được thực hiện có màu xanh lá và liền nét.

Trạng thái không thực hiện có dạng đường đứt nét.

* Cách mở các hộp thoại trong cửa sổ mô phỏng:

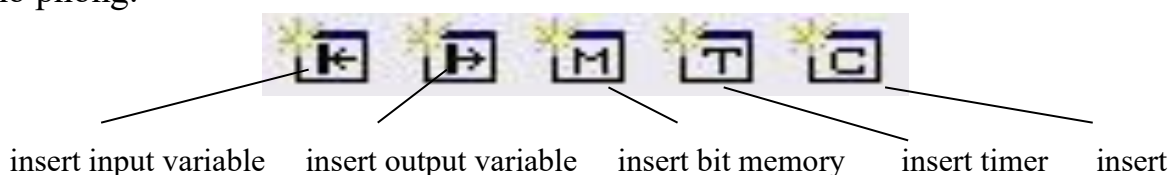
- Hộp thoại các biến vào: click chuột vào biểu tượng “insert input variable” trong cửa sổ mô phỏng.

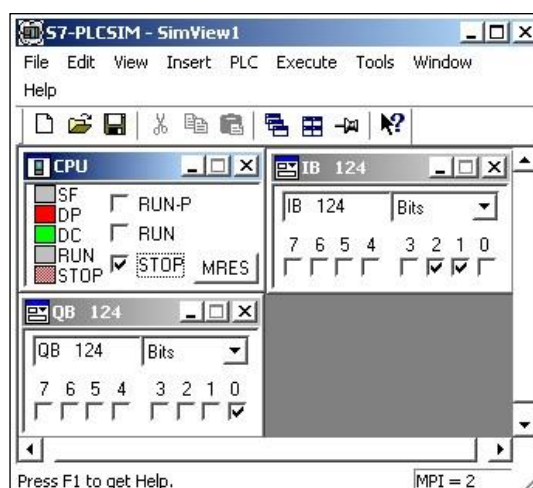
- Hộp thoại các biến ra: click chuột vào biểu tượng “insert output variable” trong cửa sổ mô phỏng.

- Hộp thoại các ô nhớ nội: click chuột vào biểu tượng “insert bit memory” trong cửa sổ mô phỏng.

- Hộp thoại các timer: click chuột vào biểu tượng “insert timer” trong cửa sổ mô phỏng.

- Hộp thoại các counter: click chuột vào biểu tượng “insert counter” trong cửa sổ mô phỏng.





Hình 2.17: cửa sổ mô phỏng **Chú**


Ý:


- Ở chế độ kiểm tra, ta không thể thực hiện được bất kỳ sự thay đổi nào trong chương trình.

- Thông thường, khi lập trình ta thường bắt đầu từ địa chỉ vào và ra ở byte 0 (I0.0, I0.1,...Q0.0, Q0.1...), tuy nhiên khi mở các hộp thoại trong cửa sổ mô phỏng thì chúng thường có địa chỉ mặc định nên khi gọi cửa sổ mô phỏng thì phải sửa lại các biến vào/ra trên bảng mô phỏng cho phù hợp.

5. Download chương trình lên PLC.

Sau khi chương chạy đúng ý đồ lập trình, ta có thể cho chạy trên PLC thật, thao tác được thực hiện trình tự như sau:

- Để download chương trình về CPU thì phải tắt chức năng mô phỏng.
- Kết nối máy tính với PLC bằng cáp nối.
- Download chương trình về PLC: trên cửa sổ lập trình ta click nút  (hoặc menu **PLC/Download**).

- **Ta xem hoạt động chương trình bằng cách mở chức năng quan sát** - click vào biểu tượng mắt kính  trên thanh toolbar **Lưu ý:**

- Download phần cứng: trong cửa sổ HW Config, nhấp biểu tượng download/ok/ok.
- Download chương trình: trong cửa sổ lập trình, nhấp biểu tượng download/ok/ok.

B. PHẦN THỰC HÀNH

1. Bài tập 1:

Hãy mở chương trình, khởi tạo một project và thiết lập phần cứng cho project này theo các module hiện có tại xưởng thực hành.

- **Bước 1:** Nhấp vào **Start/Simatic/Simatic Manager** hoặc nhấp đôi vào biểu tượng Step 7 trên màn hình để mở chương trình.

- **Bước 2:** Chọn loại CPU đúng với thiết bị hiện có (xem các thông số trên thiết bị tại xưởng).

- **Bước 3:** Chọn khối cần lập trình (Ví dụ khối OB1) và chọn ngôn ngữ lập trình (ví dụ ngôn ngữ LAD) (hình 3) sau đó click **Next**.

- **Bước 4:** Đặt tên cho project sau đó click **Finish**.

- **Bước 5:** trong lại cửa sổ SIMATIC Manager, chọn SIMATIC 300 (bên trái cửa sổ)/ nhấp đôi vào biểu tượng Hardware bên phải cửa sổ để mở cửa sổ HW Config.

- **Bước 6:** Thiết lập phần cứng cho CPU

- **Bước 7:** Thiết lập phần cứng cho các module mở rộng:

2. Bài tập 2:

Hãy mở project vừa khởi tạo bên trên, viết chương trình điều khiển đơn giản theo hướng dẫn của giáo viên, sau đó cho chạy mô phỏng chương trình này.

- **Bước 1:** Mở phần mềm Step

- **Bước 2:** Nhấp cancel hoặc đóng cửa sổ “New project”.

- **Bước 3:** vào Menu File/chọn open hoặc nhấp vào biểu tượng Open Project/Library, lập tức xuất hiện hộp thoại Open Project.

- **Bước 4:** Trong hộp thoại Open Project, chọn tên file, nhấp OK.

- **Bước 5:** Mở OB1 để vào cửa sổ lập trình.

- **Bước 6:** Tiến hành lập trình và lưu chương trình.

- **Bước 7:** Gọi cửa sổ mô phỏng.

- **Bước 8:**Download chương trình.

- **Bước 9:** Kích hoạt chức năng kiểm tra và quan sát.

- **Bước 10:** Chạy chương trình mô phỏng: trên bảng CPU ta click nút **RUN**.

- **Bước 11:** Nhập tín hiệu vào mô phỏng

Bài 3: CÁC PHÉP TOÁN NHỊ PHÂN

Mục tiêu của bài:

- Trình bày được các chức năng, phạm vi ứng dụng của các phép toán nhị phân như: lệnh vào/ra tiếp điểm, các liên kết nhị phân, đại số Boolean, lệnh Set, Reset, lệnh nhận viết cạnh tín hiệu...
- Ứng dụng linh hoạt các chức năng của phép toán nhị phân.
- Thao tác lập trình và mô phỏng chính xác, chương trình ngắn gọn, dễ hiểu.

Nội dung của bài:

I. Các lệnh vào/ra tiếp điểm

A. PHẦN LÝ THUYẾT:

- Các lệnh vào là các lệnh do người điều khiển tác động, cài đặt hoặc được lấy từ các ngõ ra của các cảm biến.... Trong S7-300, các lệnh vào thường có địa chỉ chứa trong 1 bit, được đặt tên gồm có 2 phần, phần chữ và phần số (I0.0, I0.1, ...). Phần chữ luôn có ký tự I, phần số có hai phần: phần trước dấu chấm là byte và sau dấu chấm là bit chứa địa chỉ đó.

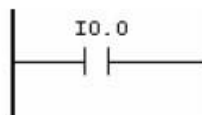
- Các lệnh ra là các lệnh được xuất ở các ngõ ra sau khi PLC đã xử lý các tín hiệu vào, đó thường là kết quả mà người lập trình mong muốn. Trong S7-300, các lệnh ra thường có địa chỉ 1 bit, địa chỉ ngõ ra cũng gồm có 2 phần như địa chỉ vào (Q0.0, Q0.1, ...). Phần chữ luôn có ký tự Q, phần số cũng có hai phần xác định byte và bit chứa địa chỉ ra.

- Trong PLC S7-300, mức logic thấp (mức 0) có điện thế 0V, mức cao (mức 1) có điện thế 24V.

1. Các lệnh vào:

* Tiếp điểm thường hở:

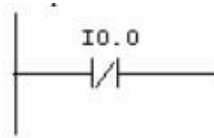
- Ký hiệu:



- Nguyên lý làm việc: Đây là tiếp điểm thường mở, nếu giá trị của bit chứa địa chỉ là 0 (I0.0=0) thì tiếp điểm ở trạng thái hở. Ngược lại, nếu giá trị bit đó là 1 thì tiếp điểm đóng lại, khi tiếp điểm đóng thì cho dòng điện đi qua.

* Tiếp điểm thường đóng, sẽ được mở nếu I0.0 = 1

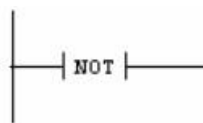
- Ký hiệu:



- Nguyên lý làm việc: Đây là tiếp điểm thường đóng, nếu giá trị của bit chứa địa chỉ là 0 ($I0.0=0$) thì tiếp điểm ở trạng thái đóng, cho dòng điện đi qua. Ngược lại, nếu giá trị bit đó là 1 thì tiếp điểm ở trạng thái hở, khi tiếp điểm hở thì ngắt dòng điện đi qua.

*** Lệnh Not:**

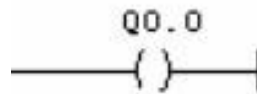
- Ký hiệu:



Nguyên lý làm việc: Lệnh NOT làm đảo mức tín hiệu trước đó từ mức cao thành mức thấp và ngược lại. Nếu $KT=1$ thì $KQ=0$; Nếu $KT=0$ thì $KQ=1$

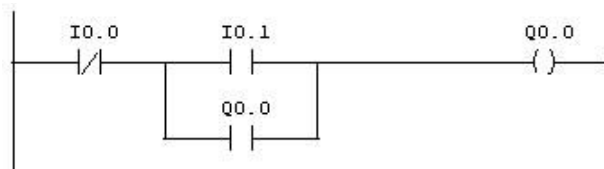
2. Lệnh ra:

- Ký hiệu:



- Nguyên lý làm việc: Hoạt động giống như cuộn dây của rơle (công tắc tơ), nếu có dòng điện đi qua cuộn dây thì bit chứa địa chỉ ngõ ra có giá trị bằng 1 ($Q0.0=1$), đồng thời các tiếp điểm thuộc cuộn dây cũng được tác động (tiếp điểm thường mở được đóng lại và tiếp điểm thường đóng bị mở ra). Lưu ý là các tiếp điểm thuộc cuộn dây phải có địa chỉ trùng với địa chỉ cuộn dây thì chương trình mới chạy đúng. Ngược lại khi không có dòng điện đi qua thì giá trị bit chứa địa chỉ bằng 0.

Thí dụ: Ta viết chương trình sau:



Chương trình trên có:

- 1 tiếp điểm thường đóng được xác định tại địa chỉ vào I0.0
- 1 tiếp điểm thường mở được xác định tại địa chỉ vào I0.1
- 1 cuộn dây ngõ ra được xác định tại địa chỉ ra Q0.0

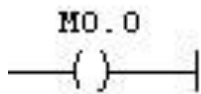
- 1 tiếp điểm thường mở được xác định tại địa chỉ Q0.0, đây là tiếp điểm phụ thuộc cuộn dây Q0.0, nó sẽ được tác động (đóng lại) nếu có dòng điện qua cuộn dây.

Nguyên lý làm việc của chương trình: Khi $I0.1 = 1$ (tác động 24VDC vào ngõ I0.1) thì tiếp điểm tại địa chỉ này đóng lại, cho dòng điện đi qua cuộn Q0.0, làm đóng tiếp điểm thường mở Q0.0. Khi đó nếu I0.1 bị đặt xuống mức thấp thì dòng điện vẫn duy trì qua cuộn dây do có tiếp điểm tự giữ. Khi cuộn dây đang có dòng điện, nếu tác động cho $I0.0=1$ thì tiếp điểm tại địa chỉ này mở ra làm ngắt dòng điện qua cuộn dây Q0.0, nghĩa là tiếp điểm tự giữ Q0.0 cũng bị ngắt. Nếu I0.0 bị đặt xuống mức thấp trở lại thì cuộn dây cũng bị ngắt điện.

3. Bit nhớ trung gian:

Bit nhớ trung gian hỗ trợ rất đắc lực cho cho việc lập trình khi muốn dùng cuộn dây mà không muốn xuất tín hiệu ngõ ra. Vì số ngõ ra trên PLC là có giới hạn nên việc dùng các bit nhớ trung gian trở nên hết sức hiệu quả cho việc điều khiển. Hoạt động của một bit nhớ trung gian hoàn toàn giống với cuộn dây role: Khi có dòng điện đi qua cuộn dây thì các tiếp điểm thuộc role sẽ bị tác động, tuy nhiên tín hiệu sẽ không xuất ra ở bất kỳ ngõ ra nào trên PLC.

- Ký hiệu:



- Nguyên lý hoạt động: khi có dòng điện đi qua cuộn dây M0.0 thì bit chứa địa chỉ M0.0 = 1, đồng thời các tiếp điểm thuộc cuộn dây cũng được tác động.

Công dụng của bit nhớ trung gian sẽ được minh họa rõ hơn trong phần bài tập áp dụng của các bài học tiếp theo.

B. PHẦN THỰC HÀNH:

4. Bài tập áp dụng:

* **Bài tập 1:** Viết chương trình mở máy trực tiếp động cơ không đồng bộ 3 pha.

* **Bài tập 2:** Viết chương trình điều khiển mở máy động cơ không đồng bộ 3 pha, điều khiển 2 nơi.

* **Bài tập 3:** Viết chương trình điều khiển mở máy 3 động cơ không đồng bộ 3 pha tắt-mở độc lập.

* **Bài tập 4:** Viết chương trình điều khiển 3 động cơ không đồng bộ 3 pha theo yêu cầu sau:

- Ba động cơ phải được khởi động và dừng tuần tự: Nhấn ON 1, Đ1 chạy, nhấn ON 2, Đ2 chạy, nhấn ON3, Đ3 chạy. Khi dừng máy thì theo trình tự ngược lại (có 3 nút OFF) và có nút dừng khẩn cấp.

* **Bài tập 5:** Viết chương trình điều khiển 1 động cơ không đồng bộ 3 pha roto lồng sóc quay 2 chiều, khởi động Y/ Δ , dùng các lệnh vào/ra cơ bản.

* **Bài tập 6:** Viết chương trình điều khiển mạch đèn hầm theo nguyên tắc: nhấn ON1, đèn 1 sáng, nhấn ON2, đèn 2 sáng, đèn 1 tắt, nhấn ON3, đèn 3 sáng, đèn 2 tắt... tương tự cho đến đèn 6, khi đi theo chiều ngược lại thì nhấn ON5, đèn 5 sáng, đèn 6 tắt, nhấn ON4, đèn 4 sáng, đèn 5 tắt ... tương tự đến khi nhấn OFF để tắt toàn hệ thống.

Các bước thực hiện:

- **Bước 1:** Mở chương trình, khởi tạo project.
- **Bước 2:** Mở cửa sổ lập trình.
- **Bước 3:** Tiến hành lập trình theo yêu cầu bài tập.
- **Bước 4:** Lưu chương trình.
- **Bước 5:** Mô phỏng chương trình.
- **Bước 6:** Hiệu chỉnh chương trình.
- **Bước 7:** Mô phỏng lại chương trình mới vừa hiệu chỉnh - **Bước 8:** Lưu chương trình sau khi hiệu chỉnh.

C. PHẦN MỞ RỘNG:

Thiết lập phần cứng theo tên:

* Khi viết một chương trình phức tạp có nhiều ngõ vào/ra, để dễ quản lý chương trình, ta thường đặt tên mang ý nghĩa gợi nhớ cho các ngõ vào ra, đây thường được gọi là cách thiết lập phần cứng theo tên và được thực hiện như sau:

- Mở cửa sổ HW Config.
- Nhấp phải tại module cần thiết lập/chọn Edit Symbols, lập tức xuất hiện hộp thoại Edit Symbols
- Đặt tên cho các ngõ vào/ra tại cộng Symbol/Apply/Ok.

Khi cần thay đổi tên cho một ngõ vào/ra thì vào lại hộp thoại Edit Symbols để thay đổi.

Khi muốn bỏ tên cho một ngõ vào/ra ta cũng vào hộp thoại Edit Symbols/nhấp chọn hàng có tên muốn bỏ/ nhấp Delete/OK

Cách trên chỉ thực hiện được với các ngõ vào/ra mà không áp dụng cho các đối tượng khác như bit nhớ trung gian, timer, counter... Để đặt tên cho các đối tượng này ta dùng cách sau:

- Trong cửa sổ lập trình, nhấp phải tại đối tượng muốn đặt tên/chọn Edit Symbols, lập tức xuất hiện hộp thoại Edit Symbols.
- Đặt tên cho đối tượng/Apply/Ok.

- Khi cần bỏ tên đối tượng ta cũng vào lại hộp thoại Edit Symbols theo cách trên/chọn cả dòng chứa đối tượng/nhấp Delete/Ok.

Lưu ý là cách này có thể áp dụng cho mọi đối tượng nhưng mỗi lần chỉ thực hiện được trên một đối tượng.

II. Các liên kết nhị phân, đại số boolean.

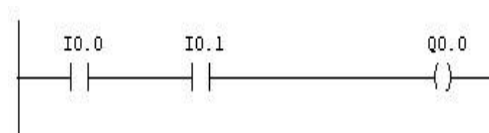
A. PHẦN LÝ THUYẾT:

1. Phép toán AND:

$$X \text{ AND } Y = X \times Y = Z$$

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Thí dụ: I0.0 x I0.1 = Q0.0

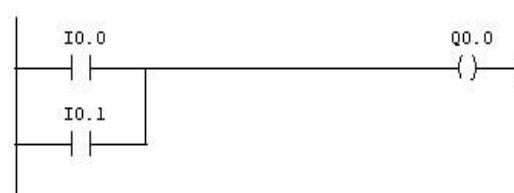


2. Phép toán OR:

$$X \text{ OR } Y = X + Y = Z.$$

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Thí dụ: I0.0 + I0.1 = Q0.0



Ngoài ra S7-300 còn có các lệnh AND double word, OR double word.

B. PHẦN THỰC HÀNH

3. Bài tập áp dụng:

Viết chương trình điều khiển các động cơ hoạt động theo yêu cầu sau:

- Khi nhấn ON1 và ON2 thì động cơ 1 chạy.
 - Nhấn ON1 hoặc ON2 thì động cơ 2 chạy.
 - Nhấn OFF để dừng toàn hệ thống trước khi cho hoạt động lại.
- (Viết chương trình cho trường hợp dùng SW chuyển mạch và dùng nút nhấn)

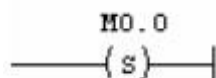
Các bước thực hiện:

- **Bước 1:** Mở chương trình, khởi tạo project.
- **Bước 2:** Mở cửa sổ lập trình.
- **Bước 3:** tiến hành lập trình theo yêu cầu bài tập.
- **Bước 4:** Lưu chương trình.
- **Bước 5:** Mô phỏng chương trình.
- **Bước 6:** Hiệu chỉnh chương trình.
- **Bước 7:** Mô phỏng lại chương trình mới vừa hiệu chỉnh
- **Bước 8:** Lưu chương trình sau khi hiệu chỉnh.

III. Lệnh Set/Reset

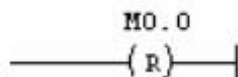
A. PHẦN LÝ THUYẾT:

1. Lệnh Set(S)

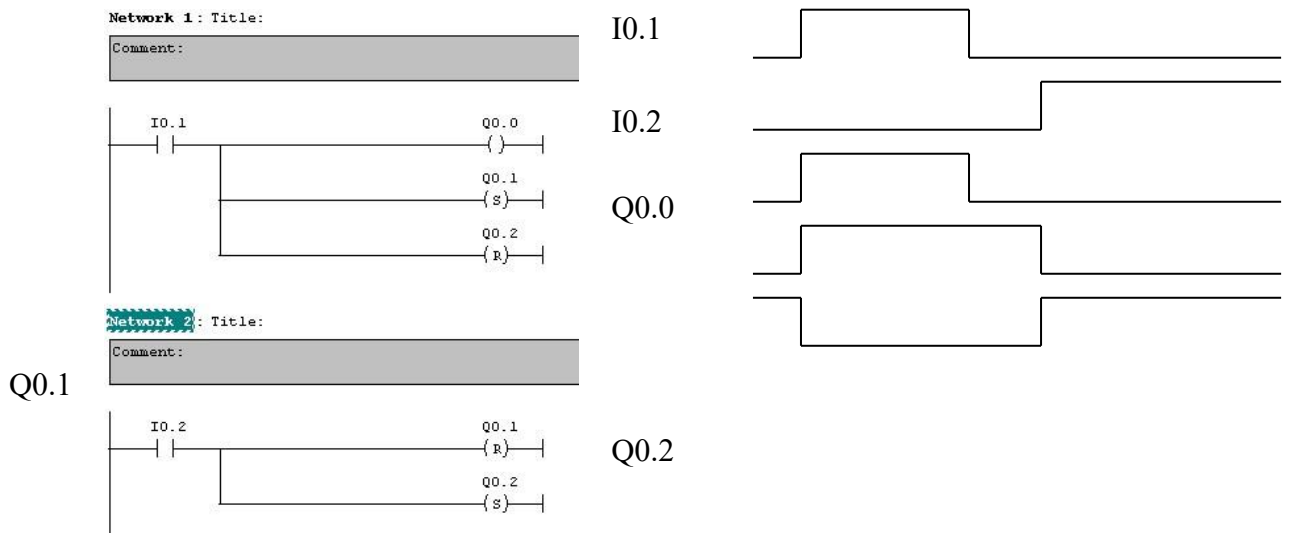


SET: Đặt giá trị của Bit cần điều khiển lên 1 khi được cấp dòng điều khiển và được duy trì cho đến khi nó bị Reset bởi một lệnh khác.

2.Lệnh Reset(S)



RESET: Đặt giá trị của Bit cần điều khiển xuống 0 khi được cấp dòng điều khiển và duy trì trạng thái này cho đến khi nó bị Reset bằng một lệnh khác. **Ví dụ:** mô tả các lệnh vào ra và S, R :



Hình 3.1: Mô tả các lệnh vào ra và S, R

Giải thích dạng sóng: Khi I0.1=1 thì các ngõ ra Q0.0 và Q0.1 lên 1, Q0.2 bị reset nên xuống mức 0. Khi I0.1 chuyển trạng thái từ 1 sang 0 thì các lệnh Set/Reset không đổi trạng thái mà chỉ có ngõ ra Q0.0 chuyển trạng thái bằng 0. Lúc này nếu muốn Q0.1=0 và Q0.2=1 thì phải tác động cho I0.2=1. Như vậy lệnh Set/Reset là lệnh có nhớ nên không cần duy trì ngõ vào ở mức tích cực.

3. Lệnh Set/Reset một FlipFlop:

Flip Flop : Một Flip Flop có một ngõ vào Set & một ngõ vào Reset, Bit nhớ được Set hoặc Reset phụ thuộc vào ngõ nào có RLO =1. Và nếu cả 2 ngõ đều có RLO = 1 thì cần xét sự ưu tiên.

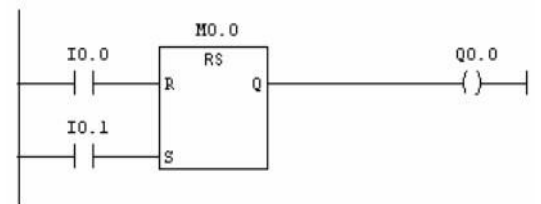
* Lệnh RS: RS Flip Flop ưu tiên Set.

Nếu I0.0=1 , I0.1=0 thì M0.0=0, Q0.0=0.

Nếu I0.0=0 ,I0.1=1 thì M0.0=1 ,Q0.0=1

Nếu I0.0=I0.1=0 Thì không có gì thay đổi.

Nếu I0.0=I0.1=1 thì M0.0=Q0.0=1



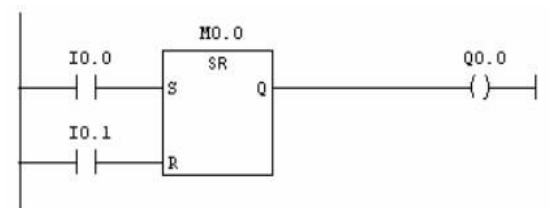
* Lệnh SR: Flip Flop ưu tiên Reset.

Nếu I0.0=1 , I0.1=0 thì M0.0=1, Q0.0=1

Nếu I0.0=0 ,I0.1=1 thì M0.0=0 ,Q0.0=0

Nếu I0.0=I0.1=0 Thì không có gì thay đổi.

Nếu I0.0=I0.1=1 thì M0.0=Q0.0=0.



B. PHẦN THỰC HÀNH

4. Bài tập áp dụng:

Sử dụng lệnh Set/Reset để viết các chương trình sau:

* **Bài tập 1:** Viết chương trình mở máy trực tiếp động cơ không đồng bộ 3 pha.

* **Bài tập 2:** Viết chương trình điều khiển 3 động cơ không đồng bộ 3 pha theo yêu cầu sau:

- Ba động cơ phải được khởi động và dừng tuần tự: Nhấn ON 1, Đ1 chạy, nhấn ON 2, Đ2 chạy, nhấn ON3, Đ3 chạy. Khi dừng máy thì theo trình tự ngược lại (có 3 nút OFF) và có nút dừng khẩn cấp.

- Thiết lập phân cứng theo tên.

* **Bài tập 3:** Viết chương trình điều khiển 1 động cơ không đồng bộ 3 pha roto lồng sóc quay 2 chiều, khởi động Y/ Δ , dùng các lệnh SR.

* **Bài tập 4:** Viết chương trình điều khiển mạch đèn hầm theo nguyên tắc: nhấn ON1, đèn 1 sáng, nhấn ON2, đèn 2 sáng, đèn 1 tắt, nhấn ON3, đèn 3 sáng, đèn 2 tắt... tương tự cho đến đèn 4, khi đi theo chiều ngược lại thì nhấn OFF4, đèn 4 tắt, đèn 3 sáng, nhấn OFF3, đèn 3 tắt, đèn 2 sáng ... tương tự đến khi nhấn OFF1 để tắt toàn hệ thống.

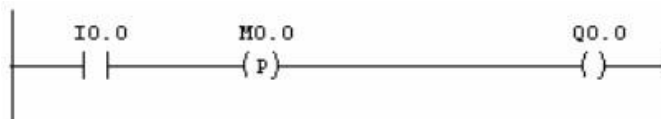
IV. Lệnh nhận biết cạnh tín hiệu:

A. PHẦN LÝ THUYẾT:

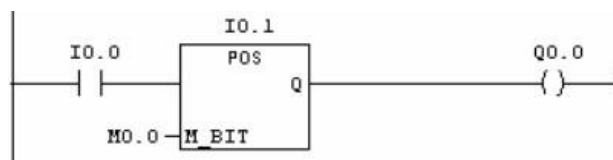
1. Nhận biết tín hiệu cạnh lên (Vi phân cạnh lên) :

M0.0 lưu giá trị KQ ở vòng quét trước

Khi I0.0 chuyển trạng thái từ 0 sang 1 và M0.0 =0 thì Q0.0 =1



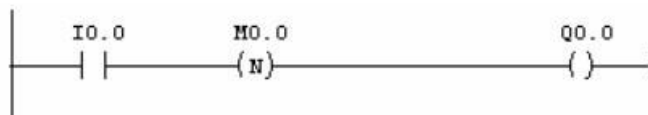
- **Lệnh POS:** Khi I0.0=1 và I0.1 chuyển trạng thái từ 0 lên 1 thì Q0.0 ON trong 1 chu kì, hay nói cách khác, Q0.0 chỉ ON tại thời điểm thoả điều kiện bài toán.



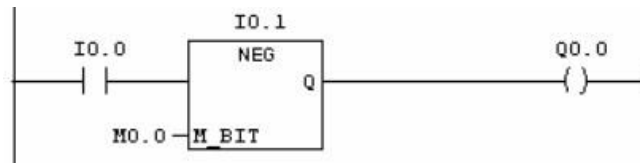
2. Nhận biết tín hiệu cạnh xuống (Vi phân cạnh xuống):

M0.0 lưu giá trị KQ ở vòng quét trước

Khi I0.0 chuyển trạng thái từ 1 xuống 0 và M0.0=1 thì Q0.0=1



Lệnh NEG: Khi I0.0=1 và I0.1 chuyển trạng thái từ 1 xuống 0 thì Q0.0 ON trong 1 chu kì, hay nói cách khác Q0.0 chỉ ON tại thời điểm thoả điều kiện bài toán.



Như vậy trong cả 2 lệnh vi phân cạnh xuống và vi phân cạnh lên thì Q0.0 chỉ ON trong 1 chu kì tại thời điểm thoả điều kiện.

3.Lệnh Save :

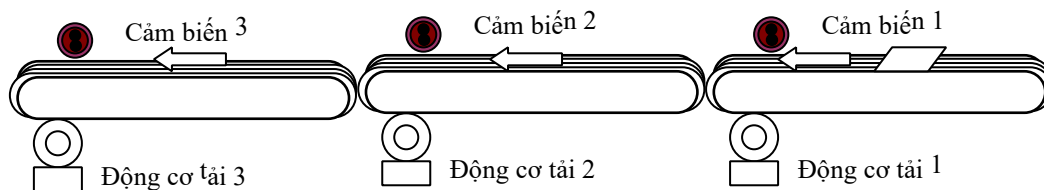
Lưu giá trị RLO (KQ) vào Bit cờ BR (Binary Result Bit)



B. PHẦN THỰC HÀNH:

4. Bài tập áp dụng:

Viết chương trình điều khiển hệ thống 3 băng tải như hình vẽ sau:



Hình 3.2: Hệ thống 3 băng tải.

- Khi nhấn nút Start, hệ thống bắt đầu hoạt động, băng tải 1 chạy. - Khi cảm biến 1 phát hiện tấm kim loại thì băng tải 2 chạy.
- Khi cảm biến 2 phát hiện tấm kim loại thì băng tải 3 chạy
- Khi tấm kim loại ra khỏi vùng phát hiện của cảm biến 2 và tải 3 chạy thì tải 2 dừng.
- Khi tấm kim loại ra khỏi vùng phát hiện của cảm biến thì tải 3 dừng.
- Nhấn Stop hệ thống dừng khẩn cấp

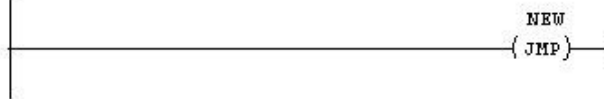
V. Lệnh nhảy

A. PHẦN LÝ THUYẾT:

1. Lệnh nhảy không điều kiện:

Network 3 : Cõi lệnh Nhảy không điều kiện

- Cõi lệnh nhảy tới NEW



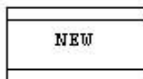
Network 4 : Title:

Comment:



Network 5 : Vị trí lệnh JUMP " NEW " đến

- Cõi label "NEW"
- Chương trình



Lệnh nhảy JMP: Nhãn nhảy có thể có tới 4 ký tự, ký tự đầu tiên phải là một chữ cái hoặc ký tự "-". Nhãn nhảy đánh dấu điểm tiếp tục làm việc của chương trình. Bất kỳ lệnh nhảy và điểm nhảy tới phải ở trong một khối (Độ dài lớn nhất của lệnh nhảy = 64kbyte). Điểm nhảy tới chỉ xuất hiện một lần trong khối.

Lệnh nhảy có thể sử dụng trong OB, FB và FC.

Chèn nhãn nhảy: program Elements / Logic control / JUMP / Label.

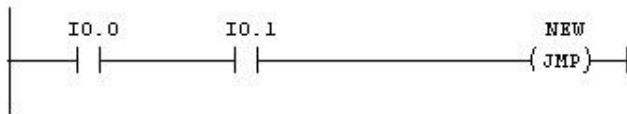
JMP :Một lệnh nhảy không điều kiện làm cho việc xử lý chương trình nhảy đến nhãn nhảy bất chấp RLO.

Chú ý : Tên nhãn phải giống nhau và phân biệt chữ hoa & chữ thường.

2. Lệnh nhảy có điều kiện.

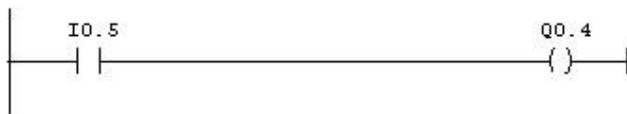
Network 3: Gõ lệnh Nhảy có điều kiện

```
- Gõ lệnh nhảy tới NEW
```



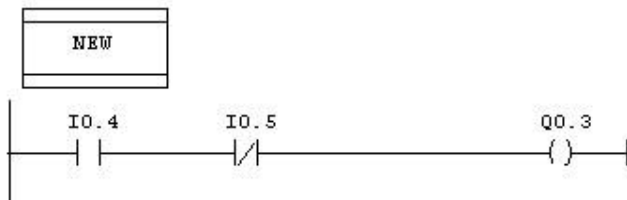
Network 4: Title:

```
Comment:
```



Network 5: Vị trí lệnh JUMP "NEW" đến

```
- Gõ label "NEW"  
- Chương trình ....
```



JMP: Nhảy có điều kiện “JMP” chỉ nhảy được thi hành nếu RLO = “1”. Ngoài ra còn có lệnh “JMPN” = JUMP NOT được thực hiện khi RLO = “0”.

B. PHẦN THỰC HÀNH:

3. Bài tập ứng dụng

Dùng lệnh nhảy để viết chương trình điều khiển một động cơ quay 2 chiều hoạt động theo yêu cầu sau:

- Khi muốn động cơ khởi động Y/ Δ , đảo chiều gián tiếp thì nhấn “START”, sau đó nhấn “ON T” hoặc “ON N”.

- Khi muốn động cơ khởi động trực tiếp (chạy chế độ Δ), đảo chiều trực tiếp thì nhấn “ON T” hoặc “ON N”.

- Nhấn “OFF” để dừng động cơ.

Bài 4: CÁC PHÉP TOÁN SỐ HỌC

Mục tiêu của bài:

- Trình bày được chức năng, phạm vi ứng dụng các phép toán số.
- Ứng dụng linh hoạt các chức năng của phép toán số.
- Thao tác lập trình và mô phỏng chính xác, chương trình ngắn gọn, dễ hiểu.

Nội dung của bài:

I. Các lệnh Timer:

A. PHÂN LÝ THUYẾT

1. Timer trong PLC:

Timer là bộ tạo thời gian trễ giữa tín hiệu vào và tín hiệu ra nên trong điều khiển thường được gọi là khâu trễ. Các công việc điều khiển cần nhiều chức năng Timer khác nhau. Trong PLC, một Word (16bit) trong vùng dữ liệu được gán cho một trong các Timer.

Một Timer có các ngõ vào và ngõ ra tương ứng như sau:

Ngõ vào Start (bắt đầu): Timer được bắt đầu với sự thay đổi tín hiệu từ mức “0” lên mức “1” ở ngõ vào Start.

+ Thời gian đặt cho Timer và hoạt động của Timer (có nhiều loại Timer khác nhau) phải được lập trình trước khi chương trình thực hiện.

+ Đặt thời gian theo cú pháp: S5T#1min30s, dữ liệu này được lưu trong ô nhớ 16 bit

- **Ngõ vào Reset (xóa):** tín hiệu mức “1” ở ngõ vào Reset làm dừng Timer, lúc đó:

+ Thời gian hiện hành được đặt về 0 +

Ngõ ra Q của timer được xóa về “0”.

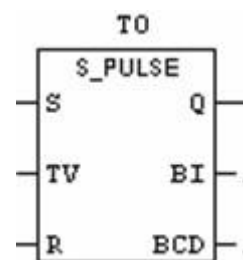
- **Các ngõ ra số:** Giá trị thời gian thực sự của Timer có thể đọc được từ ngõ ra số.

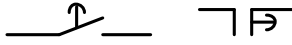


+ Ngõ ra BI: Đọc số nhị phân.

+ Đặt ra BCD: Đọc số thập phân.

- **Ngõ ra nhị phân:** trạng thái tín hiệu ngõ ra nhị phân Q của Timer là 0 (tín hiệu ngõ ra mức thấp), hay 1 (tín hiệu ngõ ra mức cao) phụ thuộc vào từng loại Timer.

Trạng thái tín hiệu ở ngõ ra nhị phân cũng chính là trạng thái tín hiệu của bit T của Timer.

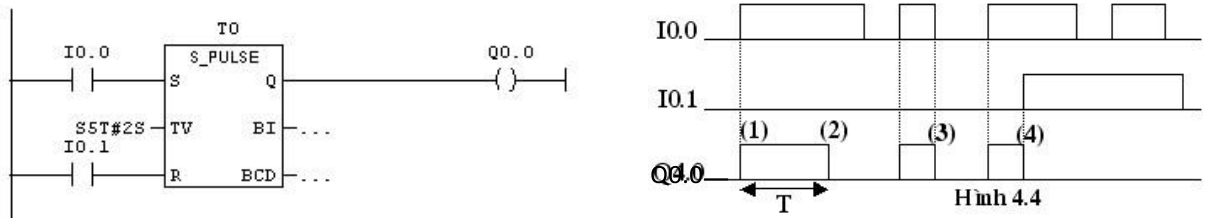


- PLC S7-300 có tổng số 128 Timer được chia thành 5 loại khác nhau:
- + Timer tạo xung không có nhớ (Pulse Timer – SP)
 - (thường mở – đóng theo thời gian chính định, không duy trì)
- + Timer tạo xung có nhớ (Extended Pulse Timer - SE)
 - (thường mở - đóng theo thời gian chính định, có duy trì)
- + Trễ theo sườn lên không có nhớ (On Delay Timer - SD)
 - 
 - (thường mở – đóng chậm)
- + Trễ theo sườn lên có nhớ (Latching On Delay Timer - SS)
 - 
 - (thường mở – đóng chậm, có “nhớ”)
- + Timer trễ theo sườn xuống (Off Delay Timer - SF)
 - 
 - (thường mở - đóng nhanh mở chậm)

2. Timer tạo xung không có nhớ (Pulse Timer – SP):

- Ngõ ra của “pulse Timer” là “1” sau khi Timer được bắt đầu (1).
- Ngõ ra bị Reset về “0” nếu:
 - Hết thời gian lập trình TV(2)
 - Nếu tín hiệu Start bị reset về “0” (3)
 - Nếu có một tín hiệu “1” đưa vào ngõ Reset của Timer (4).

Chú ý: Phải duy trì ngõ S



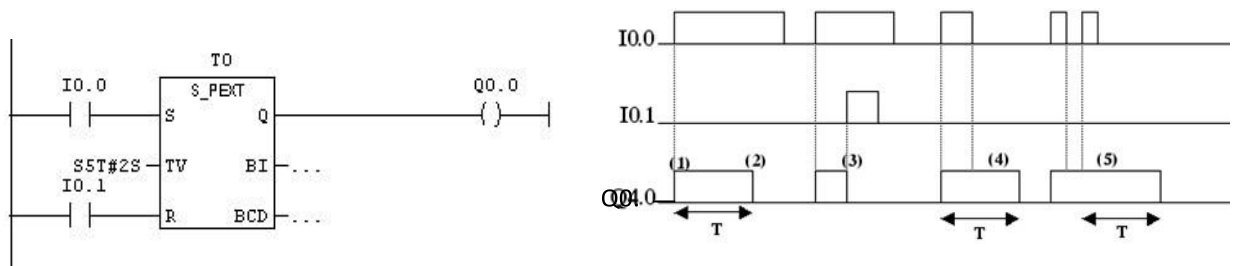
Hình 4.1: Cấu trúc lệnh S-PULSE và giản đồ xung.

Thí dụ: Viết chương trình khởi động sao/tam giác cho động cơ KĐB 3 pha dùng lệnh S-PULSE.

3. Timer tạo xung có nhớ (Extended Pulse Timer - SE)

- Ngõ ra của Extended Pulse Timer là “1” sau khi Timer được bắt đầu (1).
- Ngõ ra bị Reset về “0” nếu:
 - Quá thời gian được lập trình (2)
 - Ngõ vào Reset bị tác động (3)
- Ngõ ra Q của Timer vẫn giữ mức “1” (được chốt) ngay cả ngõ vào Start bị reset trong khi Timer đang chạy (4).

- Nếu sự thay đổi tín hiệu ở bước (1) được lập lại trong quá trình Timer đang chạy thì Timer được bắt đầu lại, nghĩa là được kích trở lại (tức là thời gian kích hoạt được tính bắt đầu trở lại) (5). **Chú ý: Không cần duy trì ngõ S**



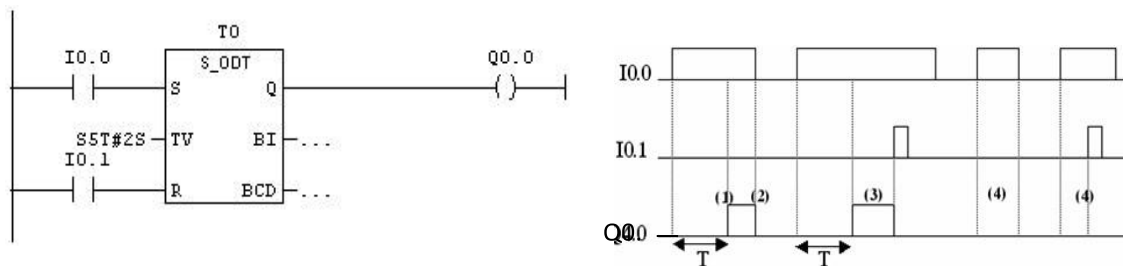
Hình 4.2: Cấu trúc lệnh S-PEXT và giản đồ xung.

Thí dụ: Viết chương trình khởi động trực tiếp một động cơ KĐB 3 pha chạy 5s rồi dừng hẳn, dùng lệnh S-PEXT.

4. Trễ theo sườn lên không có nhớ (On Delay Timer - SD)

- Ngõ ra On Delay Timer là “0” khi Timer bắt đầu.
- Nếu hết gian được lập trình và ngõ vào Start vẫn còn ở mức “1” thì ngõ ra ra On Delay Timer sẽ chuyển lên “1” (1).
- Ngõ ra bị reset về “0” nếu:
 - Ngõ vào Start bị reset (2)
 - Nếu có tín hiệu mức “1” ở ngõ vào Reset của Timer(3).

Chú ý: Phải duy trì ngõ S



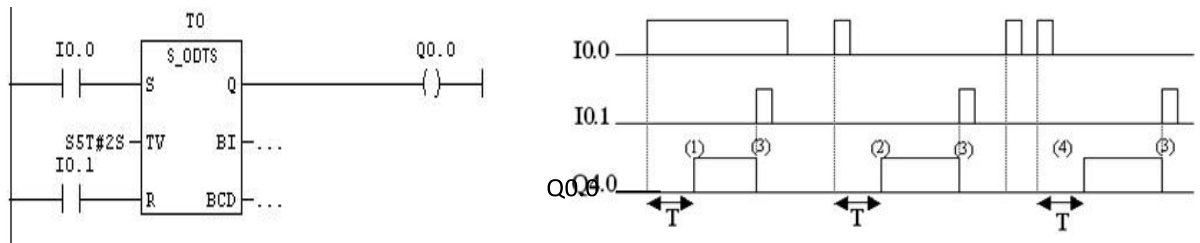
Hình 4.3: Cấu trúc lệnh S-ODT và giản đồ xung.

Thí dụ: Viết chương trình khởi động sao/tam giác cho động cơ KĐB 3 pha dùng lệnh S-ODT.

5. Trễ theo sườn lên có nhớ (Latching On Delay Timer - SS)

- Ngõ ra Latching On Delay Timer là “0” khi Timer bắt đầu.
- Nếu hết gian được lập trình thì ngõ ra ra Latching On Delay Timer sẽ chuyển lên “1” (1).

- Ngõ ra Q của Timer vẫn giữ mức “1” (được chốt) ngay cả ngõ vào Start bị reset trong khi Timer đang chạy (2).
- Ngõ ra chỉ bị Reset khi ngõ vào Reset của Timer bị tác động (3).
- Việc set và reset tiếp theo của ngõ vào Start trong khi Timer đang chạy chỉ được thực hiện khi nó bắt đầu được kích lại (4).



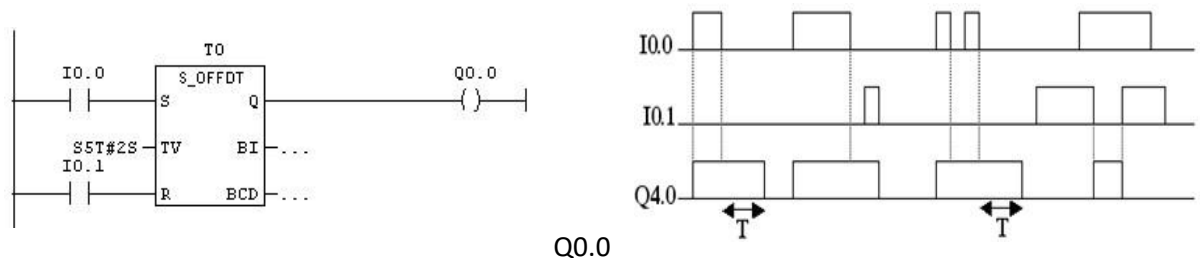
Hình 4.4: Cấu trúc lệnh S-ODTS và giản đồ xung.

Thí dụ: Dùng lệnh S-PEXT viết chương trình điều khiển hai động cơ theo yêu cầu sau:

- Nhấn ON, Đ1 chạy 5s rồi dừng hẳn - Đ2 chạy sau khi Đ1 ngừng
- Nhấn OFF để dừng khẩn cấp.

6. Timer trễ theo sườn xuống (Off Delay Timer - SF)

- Ngõ ra Q của SF được đặt lên mức “1” nếu có sự thay đổi tín hiệu từ “0” lên “1” ở ngõ vào Start.
- Nếu ngõ vào Start bị reset, Timer bắt đầu khởi động tính giờ, ngõ ra sẽ trở về “0” cho đến khi hết thời gian lập trình (2).
- Ngõ ra của SF bị reset về “0” nếu ngõ vào Reset của SF đặt lên “1” hoặc hết thời gian lập trình.



Hình 4.5: Cấu trúc lệnh S-OFFDT và giản đồ xung.

B. PHẦN THỰC HÀNH:

7. Bài tập ứng dụng

Bài tập 1: Viết chương trình điều khiển 3 động cơ khởi động tuần tự theo yêu cầu sau:

- Khi nhấn ON thì Đ1 chạy, 5s sau đến Đ2 rồi 5s sau đến Đ3.
- Khi nhấn OFF thì động cơ dừng theo tuần tự ngược lại: Đ3 dừng trước, 5s sau đến Đ2, rồi đến Đ1.
- Khi nhấn Stop thì cả 3 động cơ dừng máy đồng loạt.

* Các bước thực hành:

- Bước 1: Mở chương trình, khởi tạo Project.
- Bước 2: Lập trình mạch điều khiển ON/OFF đơn giản, chạy mô phỏng chương trình.
- Bước 3: Lập trình mạch khởi động tuần tự, chạy mô phỏng chương trình.
- Bước 4: Lập trình mạch dừng tuần tự, chạy mô phỏng chương trình.
- Bước 5: Kiểm tra và hoàn thiện chương trình.

Bài tập 2: Hãy viết chương trình điều khiển một động cơ chạy 20s, dừng 20s, chu kỳ lặp lại luân phiên, muốn dừng thì nhấn OFF.

* Các bước thực hành:

- Bước 1: Mở chương trình, khởi tạo Project.
- Bước 2: Lập trình mạch điều khiển ON/OFF đơn giản, chạy mô phỏng chương trình.
- Bước 3: Lập trình chạy 5s, dừng 5s, chạy mô phỏng chương trình.
- Bước 4: Kiểm tra và hoàn thiện chương trình.

Bài tập 3: Hãy viết chương trình điều khiển hai động cơ chạy luân phiên, mỗi động cơ chạy 20s.

* Các bước thực hành:

- Bước 1: Mở chương trình, khởi tạo Project.

Bước 2: Lập trình mạch điều khiển ON/OFF đơn giản, chạy mô phỏng chương trình.

- Bước 3: Lập trình mạch điều khiển 2 động cơ chạy luân phiên, chạy mô phỏng chương trình.
- Bước 4: Kiểm tra và hoàn thiện chương trình.

Bài tập 4: Viết chương trình điều khiển động cơ quay hai chiều, không liên động, khởi động \square/Δ , thời gian khởi động 5s.

* Các bước thực hành:

- Bước 1: Mở chương trình, khởi tạo Project.
- Bước 2: Lập trình mạch điều khiển ON/OFF đơn giản, chạy mô phỏng chương trình.
- Bước 3: Lập trình mạch điều khiển động cơ quay 2 chiều, chạy mô phỏng chương trình.
- Bước 4: Lập trình mạch khởi động \square/Δ , chạy mô phỏng chương trình.
- Bước 5: Kiểm tra và hoàn thiện chương trình.

Bài tập 5: Hãy viết chương trình điều khiển một động cơ chạy thuận 20s, dừng 10s, chạy ngược 20s, dừng 10s. Muốn dừng thì nhấn OFF.

* Các bước thực hành:

- Bước 1: Mở chương trình, khởi tạo Project.
- Bước 2: Lập trình mạch điều khiển ON/OFF đơn giản, chạy mô phỏng chương trình.
- Bước 3: Lập trình mạch điều khiển động cơ chạy thuận và dừng, chạy mô phỏng chương trình.
- Bước 4: Lập trình mạch điều khiển động cơ chạy ngược, dừng và quay trở lại chạy thuận, chạy mô phỏng chương trình.
- Bước 5: Kiểm tra và hoàn thiện chương trình.

II. Các lệnh Counter:

A. PHẦN LÝ THUYẾT

1. Nguyên lý làm việc của counter:

- Trong công nghiệp, bộ đếm rất cần cho các quá trình đếm khác nhau như: đếm số chai, đếm xe hơi, đếm số chi tiết, ...
- Một word 16bit (counter word) được lưu trữ trong vùng bộ nhớ dữ liệu hệ thống của PLC dùng cho mỗi counter.
- Số đếm được chứa trong vùng nhớ dữ liệu hệ thống dưới dạng nhị phân và có giá trị trong khoảng 0 đến 999.

Counter là bộ đếm thực hiện chức năng đếm sườn lên các xung tín hiệu đầu vào. S7 – 300 có tối đa 256 Counter (phụ thuộc CPU), ký hiệu là Cx, trong đó x

là số nguyên trong khoảng 0 –255. Trong PLC có 3 loại bộ đếm là đếm lên, ký hiệu là CU (Count Up), đếm xuống, ký hiệu là CD (Count Down) hoặc vừa đếm lên vừa đếm xuống, ký hiệu là CUD.

Thông thường bộ đếm chỉ đếm các sườn lên của tín hiệu CU và CD, song có thể mở rộng để đếm cả mức tín hiệu của chúng bằng cách sử dụng thêm tín hiệu Enable (kích đếm). Nếu có tín hiệu enable, bộ đếm sẽ đếm lên khi xuất hiện sườn lên của tín hiệu enable đồng thời tại thời điểm đó CU có mức tín hiệu 1. Tương tự bộ đếm sẽ xuống khi có sườn lên của tín hiệu Enable và tại thời điểm đó CD có mức tín hiệu 1.

Số sườn xung đếm được, được ghi vào thanh ghi 2 byte của bộ đếm gọi là thanh ghi C – word. Nội dung của thanh ghi C – word được gọi là giá trị đếm tức thời của bộ đếm và ký hiệu là CV (Current Value). Trạng thái của C – word thể hiện ở ngõ ra Q của chân C – bit. Nếu $CV \neq 0$ thì C-bit có giá trị 1. Ngược lại khi $CV = 0$, C – bit nhận giá trị logic 0. **CV luôn không âm. Bộ đếm không được đếm lùi khi CV = 0.**

Khác với Timer giá trị đặt trước PV của bộ đếm chỉ được chuyển vào C – word tại thời điểm xuất hiện sườn lên của tín hiệu (Set – S).

Bộ đếm có thể được xóa chủ động bằng tín hiệu xóa (reset). Khi bộ đếm được xóa, cả C – word và C – bit đều nhận giá trị 0.

2. Khai báo sử dụng:

Việc khai báo sử dụng một Counter bao gồm các bước:

Bước 1: Khai báo tín hiệu Enable nếu sử dụng tín hiệu chủ động kích đếm (Enable) : “Địa chỉ Bit” xác định tín hiệu sẽ được sử dụng làm tín hiệu kích cho bộ đếm. Tên của bộ đếm có dạng “Cx” với $0 \leq x \leq 255$.

Bước 2: Khai báo tín hiệu được đếm lên theo sườn lên(CU): “Địa chỉ Bit” xác định tín hiệu mà sườn lên của nó được Counter đếm. Mỗi khi xuất hiện một sườn lên của tín hiệu tại ngõ vào CU, bộ đếm sẽ tăng nội dung thanh ghi C – word (CV) lên 1 đơn vị.

Bước 3: Khai báo tín hiệu được đếm lùi theo sườn lên(CD): “Địa chỉ Bit” xác định tín hiệu mà sườn lên của nó được Counter đếm. Mỗi khi xuất hiện một sườn lên của tín hiệu tại ngõ vào CD, bộ đếm sẽ giảm nội dung thanh ghi C – word (CV) đi 1 đơn vị nếu $CV > 0$.

Trong trường hợp $CV = 0$ thì nội dung C – word không bị thay đổi.

Bước 4: Khai báo PV: Giá trị đặt trước từ (0...999) được xác định tại ngõ vào

“PV” ở dạng BCD:

o Là hằng số đếm (cú pháp khai báo: C#10, C#20)

o Qua giao tiếp dữ liệu dạng BCD. (Kiểu Word: IW, QW, MW,...)

Bước 5: Khai báo tín hiệu đặt “Set” : “Địa chỉ Bit” xác định tín hiệu mà mỗi khi xuất hiện sườn lên của nó thì hằng số PV dưới dạng BCD sẽ chuyển vào thanh ghi C- word của bộ đếm.

Bước 6: Khai báo Reset : “Địa chỉ Bit” xác định tín hiệu mà mỗi khi xuất hiện sườn lên của nó, thanh ghi C – word của bộ đếm sẽ xóa về 0. Nếu tín hiệu ở ngõ vào R = 0 thì bộ đếm không bị ảnh hưởng gì.

* Các ngõ ra:

- CV/CV_BCD : Giá trị Counter có thể là một số nhị phân hoặc số BCD được nạp vào bộ tích lũy và từ đó có thể được chuyển tới các địa chỉ khác.

- Tình trạng tín hiệu counter có thể kiểm tra tại ngõ ra “Q”:

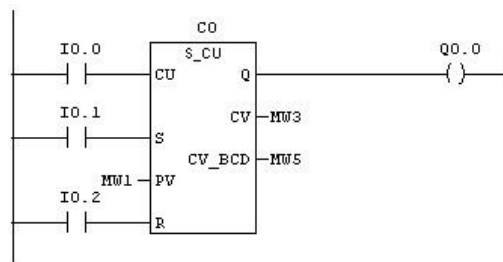
o Giá trị đếm = 0 → Q = 0.

o Giá trị đếm \neq 0 → Q = 1.

Lưu ý: - Trong các khai báo trên thì ít nhất phải có một trong hai bước 2 hoặc 3 được thực hiện.

Khi Bộ đếm bắt đầu hoạt động thì ngõ ra Q lên mức 1 và các tiếp điểm thuộc bộ đến cũng bị chuyển trạng thái.

3. Các loại bộ đếm trong PLC S7-300: * UP COUNTER:



Hình 4.6a: Cấu trúc lệnh đếm lên

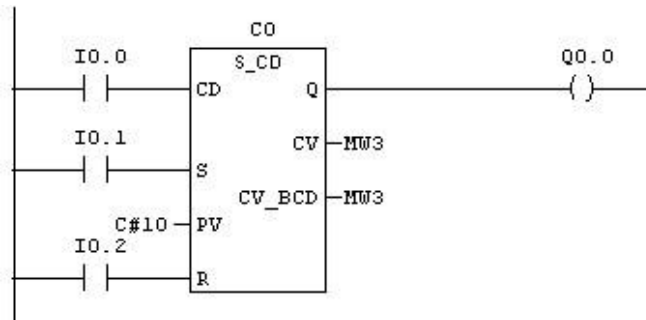
- IO.0 : Counter đếm lên khi tín hiệu này chuyển trạng thái từ 0 sang 1
- IO.1: Đặt giá trị bắt đầu và cho phép Counter đếm
- IO.2: Reset Value. Khi tín hiệu này chuyển từ 0 sang 1 thì bộ đếm bị reset, tất cả các ngõ ra đều bị reset về mức 0.

- Q0.0 = 1 khi giá trị hiện tại ở các ngõ ra CV/CV-BCD của Counter khác 0

- MW1 tại ngõ vào PV: Chứa giá trị bắt đầu đếm cho Counter, giá trị bắt đầu được ghi vào thanh ghi C- word khi ngõ vào S chuyển trạng thái từ 0 sang 1. Giá trị này có thể được đặt trực tiếp bằng số thập phân hoặc gián tiếp qua ô nhớ nội kiểu word. Đối với bộ đếm lên, ta không cần đặt giá trị ban đầu nếu số đếm bắt đầu từ 0.

- MW3, MW5: chứa giá trị đếm hiện tại dạng nhị phân và BCD.

*** DOWN COUNTER:**



Hình 4.6b: Cấu trúc lệnh đếm xuống

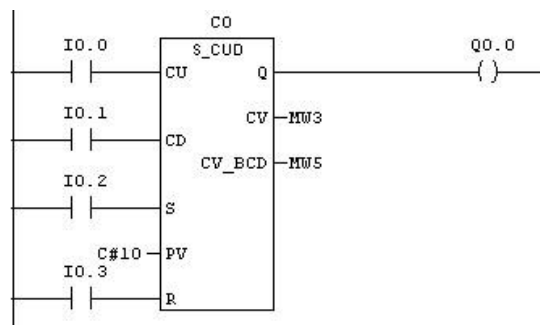
- IO.0 : Counter đếm xuống khi tín hiệu này chuyển trạng thái từ 0 sang 1
- IO.1: Đặt giá trị bắt đầu và cho phép Counter đếm
- IO.2: Reset Value. Khi tín hiệu này chuyển từ 0 sang 1 thì bộ đếm bị reset, tất cả các ngõ ra đều bị reset về mức 0.
- Q0.0 = 1 khi giá trị hiện tại ở các ngõ ra CV/CV-BCD của Counter khác 0
- Ngõ vào PV: Chứa giá trị bắt đầu đếm cho Counter, giá trị bắt đầu được ghi vào thanh ghi C- word khi ngõ vào S chuyển trạng thái từ 0 sang 1. Giá trị này có thể được đặt trực tiếp bằng số thập phân hoặc gián tiếp qua ô nhớ nội kiểu word.

Đối với bộ đếm xuống, ta phải đặt giá trị ban đầu vì bộ đếm xuống không thể hoạt động khi giá trị đến hiện tại bằng 0.

- MW3, MW5: chứa giá trị đếm hiện tại dạng nhị phân và BCD.

VD: Bộ đến C0 sẽ đếm xuống và dừng khi đủ 10 xung xuất hiện tại ngõ vào CD.

*** UP - DOWN COUNTER**



Hình 4.6c: Lệnh đếm lên

- I0.0 : Counter đếm lên khi tín hiệu này chuyển trạng thái từ 0 sang 1
- I0.1 : Counter đếm xuống khi tín hiệu này chuyển trạng thái từ 0 sang 1
- I0.2: Đặt giá trị bắt đầu và cho phép Counter đếm
- I0.3: Reset Value. Khi tín hiệu này chuyển từ 0 sang 1 thì bộ đếm bị reset, tất cả các ngõ ra đều bị reset về mức 0.
- Q0.0 = 1 khi giá trị hiện tại ở các ngõ ra CV/CV-BCD của Counter khác 0
- Ngõ vào PV: Chứa giá trị bắt đầu đếm cho Counter.
- MW3, MW5: chứa giá trị đếm hiện tại dạng nhị phân và BCD.

VD: Bộ đếm C0 sẽ đếm xuống hoặc lên tùy thuộc tín hiệu vào tại ngõ CU hay CD. Giá trị đếm bắt đầu từ 10, nếu đếm xuống thì bộ đếm sẽ dừng khi giá trị đếm hiện tại bằng 0.

B. PHẦN THỰC HÀNH

4. Bài tập ứng dụng:

Bài tập 1: Viết chương trình điều khiển một động cơ kéo băng tải có cảm biến đếm sản phẩm bên trên theo yêu cầu sau:

- Khi nhấn ON thì động cơ chạy, khi đủ 10 sản phẩm đi qua thì dừng băng tải.
- Khi nhấn OFF thì hệ thống dừng khẩn cấp.

* Các bước thực hành:

- Bước 1: Mở chương trình, khởi tạo Project.
- Bước 2: Lập trình mạch điều khiển ON/OFF đơn giản, chạy mô phỏng chương trình.
- Bước 3: Lập trình mạch cảm biến đếm sản phẩm, chạy mô phỏng chương trình.
- Bước 4: Kiểm tra và hoàn thiện chương trình.

Bài tập 2: Tương tự như bài 1, nhưng có thêm các yêu cầu sau:

- Sau khi sản phẩm thứ 10 đi qua cảm biến 2s thì tải mới dừng và tự khởi động lại sau khi dừng 20s.

Khi nhấn ON thì động cơ chạy, khi đủ 10 sản phẩm đi qua thì dừng băng tải.

- Khi nhấn Stop thì hệ thống dừng khẩn cấp.
- Khi nhấn OFF thì hệ thống dừng sau khi đủ 10 sản phẩm tiếp theo.

* Các bước thực hành:

- Bước 1: Mở chương trình, khởi tạo Project.
- Bước 2: Lập trình mạch điều khiển ON/OFF đơn giản, chạy mô phỏng chương trình.
- Bước 3: Lập trình mạch cảm biến đếm sản phẩm, chạy mô phỏng chương trình.

- Bước 4: Lập trình mạch điều khiển động cơ dừng 2s sau khi SP đi qua, chạy mô phỏng chương trình.

- Bước 5: Kiểm tra và hoàn thiện chương trình.

Bài tập 3:

Hãy viết chương trình điều khiển một động cơ chạy 30s, dừng 20s, chu kỳ lặp lại 5 lần rồi ngưng hẳn, muốn chạy lại thì mở máy lại.

* Các bước thực hành:

- Bước 1: Mở chương trình, khởi tạo Project.

- Bước 2: Lập trình mạch điều khiển ON/OFF đơn giản, chạy mô phỏng chương trình.

- Bước 3: lập trình mạch điều khiển động cơ dừng và chạy luân phiên, chạy mô phỏng chương trình.

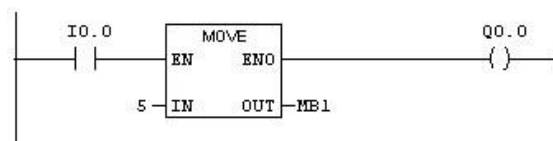
- Bước 4: Lập trình mạch đếm số lần chạy động cơ (5 lần), chạy mô phỏng chương trình.

- Bước 5: Kiểm tra và hoàn thiện chương trình.

III. Lệnh nạp và truyền dữ liệu:

A. PHẦN LÝ THUYẾT:

Các ngõ vào/ra:



Hình 4.7: Cấu trúc lệnh move

- EN: Cho phép ngõ vào, khi tín hiệu ngõ vào này I0.0=1 thì giá trị tại ngõ vào (IN) sẽ được chuyển đến ngõ ra (OUT) và lưu trong ô nhớ MB1.

- ENO: cho phép ngõ ra, tín hiệu tại đây cùng trạng thái với tín hiệu ngõ vào EN.

- IN: dữ liệu vào, có thể là số hoặc ô nhớ chứa dữ liệu.

- OUT: Dữ liệu ra, chỉ có thể là ô nhớ, nó lưu nội dung ngõ vào IN mỗi khi có tín hiệu cho phép ngõ vào. Kiểu dữ liệu giữa ngõ IN và ngõ OUT phải tương thích nhau.

- Ví dụ:

+ Nếu ngõ vào là MW thì ngõ ra cũng phải là MW hoặc MD.

+ Nếu ngõ vào là số nguyên thì ngõ ra phải là MW hoặc MD.

B. PHẦN THỰC HÀNH:

Bài tập 1: Viết chương trình điều khiển động cơ không đồng bộ 3 pha, quay hai chiều, không liên động, khởi động \square/Δ theo yêu cầu sau:

- Nhấn ON1: thời gian khởi động là 5s.
- Nhấn ON2: thời gian khởi động là 8s.
- Nhấn OFF để dừng động cơ.

* Các bước thực hành:

- Bước 1: Mở chương trình, khởi tạo Project.
- Bước 2: Lập trình mạch điều khiển ON/OFF đơn giản, chạy mô phỏng chương trình.
- Bước 3: Lập trình mạch điều khiển động cơ khởi động Y/ Δ , thời gian khởi động 5s, chạy mô phỏng chương trình.
- Bước 4: Lập trình phân truyền dữ liệu (3s hoặc 5s), chạy mô phỏng chương trình.
- Bước 5: Kiểm tra và hoàn thiện chương trình.

Bài tập 2:

Hãy viết chương trình điều khiển một động cơ theo yêu cầu sau:

- Nhấn ON1: Động cơ chạy 5s, dừng 5s, chu kỳ lặp lại 3 lần rồi ngưng hẳn.
 - Nhấn ON2: Động cơ chạy 5s, dừng 5s, chu kỳ lặp lại 6 lần rồi ngưng hẳn.
 - Nhấn Stop để dừng khẩn cấp
- * Các bước thực hành:
- Bước 1: Mở chương trình, khởi tạo Project.
 - Bước 2: Lập trình mạch điều khiển ON/OFF đơn giản, chạy mô phỏng chương trình.
 - Bước 3: Lập trình mạch điều khiển động cơ chạy và dừng luân phiên, chạy mô phỏng chương trình.
 - Bước 4: Lập trình phân truyền dữ liệu, chạy mô phỏng chương trình.
 - Bước 5: Kiểm tra và hoàn thiện chương trình.

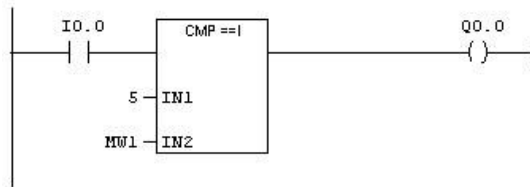
IV. Các lệnh so sánh:

A. PHẦN LÝ THUYẾT

Dữ liệu so sánh được đưa vào hai ngõ IN1 và IN2, nếu kết quả so sánh là TRUE thì bộ so sánh tích cực, tuy nhiên lúc này bộ so sánh phải được cấp nguồn thì ngõ ra mới tích cực.

TD: Viết chương trình sau:

Nếu $IN1 = IN2$ và $I0.0=1$ thì ngõ ra $Q0.0 = 1$



Hình 4.8: Cấu trúc lệnh so sánh bằng.

- Có thể dùng lệnh so sánh để so sánh các kiểu dữ liệu sau:
 - + I : So sánh những số nguyên (Dựa trên cơ sở số 16 bits)
 - + D : So sánh những số nguyên kép (Dựa trên cơ sở số 32 bits)
 - + R : So sánh những số thực (Dựa trên cơ sở số thực 32 bits)
- Có các phép so sánh như sau:
 - + == (I, D, R) : IN1 bằng IN2.(EQ)
 - + <> (I, D, R) : IN1 không bằng IN2.(NE)
 - + > (I, D, R) : IN1 lớn hơn IN2.(GT)
 - + < (I, D, R) : IN1 nhỏ hơn IN2.(LT)
 - + >= (I, D, R) : IN1 lớn hơn hoặc bằng IN2.(GE)
 - + <= (I, D, R) : IN1 nhỏ hơn hoặc bằng IN2.(LE)

B. PHẦN THỰC HÀNH:

Bài tập 1:Viết chương trình điều khiển một động cơ theo yêu cầu sau:

- Động cơ quay hai chiều (không liên động), khởi động Y/ Δ , thời gian khởi động là 5s.

- Khi nhấn nút “ON THUAN” lần đầu, động cơ chạy thuận, nhấn “ON THUAN” lần thứ hai, động cơ ngừng.

- Khi nhấn nút “ON NGUOC” lần đầu, động cơ chạy ngược, nhấn “ON NGUOC” lần thứ hai, động cơ ngừng.

* Các bước thực hành:

- Bước 1: Mở chương trình, khởi tạo Project.

- Bước 2: Lập trình mạch điều khiển ON/OFF đơn giản, chạy mô phỏng chương trình.

- Bước 3: Lập trình mạch điều khiển động cơ quay 2 chiều, chạy mô phỏng chương trình.

- Bước 4: Lập trình mạch điều khiển động cơ khởi động Y/ Δ , chạy mô phỏng chương trình.

- Bước 5: Lập trình phân đếm và so sánh để nhấn lần 2 động cơ dừng, chạy mô phỏng chương trình.

- Bước 6: Kiểm tra và hoàn thiện chương trình.

Bài tập 2:

Hãy viết chương trình điều khiển một động cơ chạy 30s, dừng 20s, chu kỳ lặp lại 5 lần rồi ngưng hẳn, muốn chạy lại thì mở máy lại, dùng bộ đếm lên.

* Các bước thực hành:

- Bước 1: Mở chương trình, khởi tạo Project.

- Bước 2: Lập trình mạch điều khiển ON/OFF đơn giản, chạy mô phỏng chương trình.

- Bước 3: Lập trình mạch điều khiển động cơ chạy và dừng luân phiên, chạy mô phỏng chương trình.

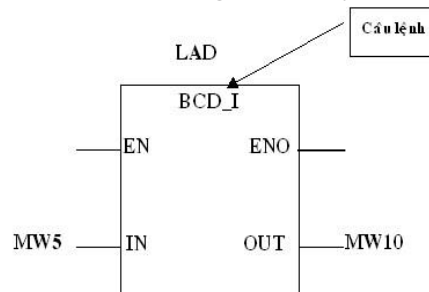
- Bước 4: Lập trình mạch điều khiển động cơ chạy 5 chu kỳ dùng bộ đếm lên và so sánh, chạy mô phỏng chương trình.

- Bước 5: Kiểm tra và hoàn thiện chương trình.

V. Các lệnh chuyển đổi dữ liệu:

Một chương trình dùng để thực hiện những chức năng toán học thì có các giá trị nhập vào bằng nút nhấn và hiển thị kết quả dạng số. Vì các chức năng toán học không thể thực hiện được ở dạng BCD do đó cần phải chuyển đổi.

Trong S7 – 300/ 400 có nhiều lệnh dùng để chuyển đổi. Tất cả những lệnh này có cùng một định dạng:



Hình 4.9: Cấu trúc lệnh chuyển dữ liệu.

- EN, ENO : khi có sự thay đổi tín hiệu từ “0” lên “1” tại ngõ vào cho phép EN thì sự chuyển đổi được thực hiện. Ngõ ra cho phép ENO luôn có tình trạng tín hiệu giống ngõ vào EN. Trường hợp không giống nhau thì nó được hướng dẫn bằng câu lệnh tương ứng.

- Ngõ vào IN: dữ liệu cần chuyển đổi. Khi EN = 1 giá trị tại IN (nội dung ô nhớ MW5) được đọc vào lệnh chuyển đổi. Đặc điểm dữ liệu:

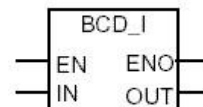
+ Có thể là hằng số hoặc ô nhớ.

+ Phải tương thích kiểu dữ liệu và kích thước ô nhớ (I, Q, M, Const, L, D...)

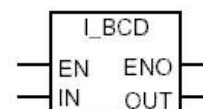
- Ngõ ra OUT : Kết quả sự truyền đổi được đưa vào địa chỉ ở ngõ ra OUT(MW10). Đặc điểm dữ liệu:

- + Chỉ có thể là ô nhớ.
- + Phải tương thích dữ liệu và kích thước ô nhớ (I, Q, M, Const, L, D...)

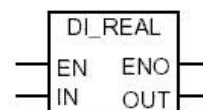
□ **BCD_I**: Chuyển đổi số nhị phân thập phân 16 bit thành số nguyên 16 bit và kết quả ghi vào OUT



□ **I_BCD**: Chuyển đổi số nguyên 16 bit IN thành số nhị phân thập phân 16 bit và kết quả ghi vào OUT



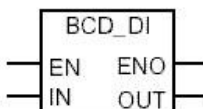
□ **DI_REAL**: Chuyển đổi số nguyên 32 bit có dấu IN thành số thực 32 bit và ghi kết quả vào OUT



□ **I_DINT**: Chuyển đổi số nguyên 16 bit thành số nguyên 32 bit và ghi kết quả vào OUT



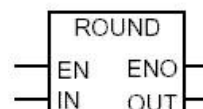
□ **BCD_DI**: Chuyển đổi số BCD thành số nguyên 32 bit và ghi kết quả vào OUT



□ **DI_BCD**: Chuyển đổi số nguyên 32 bit thành số BCD và ghi kết quả vào OUT



□ **ROUND**: Làm tròn giá trị ngõ vào thành số nguyên và ghi kết quả vào OUT



Kiểu dữ liệu và vùng nhớ của các tham số vào/ra được qui định trong bảng sau:

Tha	Kiểu dữ liệu	Vùng nhớ
EN	BOOL	I,Q,M,L,D
EN	BOOL	I,Q,M,L,D
IN	DINT	I,Q,M,L,D
OU	DWORD	I,Q,M,L,D

VI. Các lệnh toán học cơ bản:

- S7-300 có nhiều lệnh cho phép tính toán số học. Tất cả các lệnh này có cùng một định dạng.

- Ngõ vào EN: Lệnh được thực hiện nếu có sự thay đổi tín hiệu từ mức “0” lên mức “1”.

- Ngõ ra ENO: Nếu kết quả nằm ngoài phạm vi cho phép của loại dữ liệu tương ứng thì bit tràn OV và bit tràn có nhớ OS được set lên “1” và ENO = “0”. Qua đó phép tính tiếp theo qua ENO không được thực hiện.

- Ngõ vào IN1, IN2: Giá trị tại IN1 được đọc vào như toán tử thứ nhất và

IN2 như toán tử thứ 2.

Chú ý: - Phải có Sự tương thích dữ liệu và kích thước ô nhớ chứa dữ liệu. - Ngõ ra OUT: Kết quả của phép toán số học được lưu tại ngõ ra

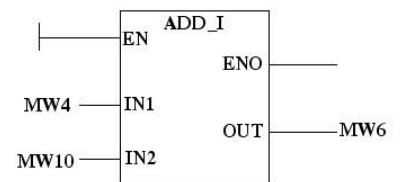
OUT Các câu lệnh:

- Cộng:

+ ADD_I : Cộng hai số nguyên.

+ ADD_DI: Cộng hai số nguyên kép.

+ ADD_R : Cộng hai số thực.

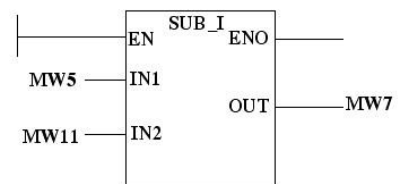


- Trừ:

+ SUB_I : Trừ hai số nguyên.

+ SUB_DI : Trừ hai số nguyên kép.

+ SUB_R : Trừ hai số thực

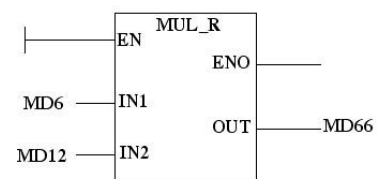


- Nhân:

+ MUL_I : Nhân hai số nguyên.

+ MUL_DI: Nhân hai số nguyên kép.

+ MUL_R : Nhân hai số thực.

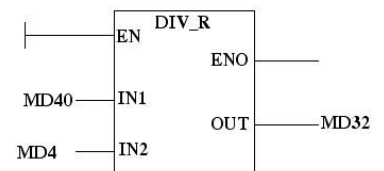


- Chia:

+ DIV_I : Chia hai số nguyên.

+ DIV_DI : Chia hai số nguyên kép.

+ DIV_R : Chia hai số thực



Thí dụ: dùng các lệnh trên để cộng, trừ, nhân chia hai số nguyên.

VII. Bài tập áp dụng các phép toán số học:

BÀI TẬP 1: Viết chương trình điều khiển động cơ không đồng bộ 3 pha hoạt động theo chế độ sau:

- Động cơ chạy thuận 30s, ngừng 10s, chạy ngược 30s, ngừng 10s.
- Sau khi cấp nguồn, nhấn nút chọn chu kỳ hoạt động, chu kỳ gồm có “3 lần”, “6 lần”, “9 lần”, động cơ sẽ chạy theo chu kỳ được chọn.
- Nhấn nút OFF để dừng khẩn cấp và bộ đếm sẽ đếm lại từ đầu khi cho hệ thống hoạt động lại.

BÀI TẬP 2: Viết chương trình điều khiển một động cơ hoạt động theo chế độ sau:

- Động cơ quay hai chiều, khởi động \square/Δ , thời gian khởi động là 5 s.
- Nhấn “ON THUAN 1” động cơ chạy thuận 30s, dừng 10s, chu kỳ lặp lại 5 lần.
- Nhấn “ON THUAN 2” động cơ chạy thuận 45s, dừng 15s, chu kỳ lặp lại 3 lần.
- Nhấn “ON NGUOC 1” động cơ chạy ngược 30s, dừng 10s, chu kỳ lặp lại 5 lần.
- Nhấn “ON NGUOC 2” động cơ chạy ngược 45s, dừng 15s, chu kỳ lặp lại 3 lần.
- Nhấn “OFF” để dừng động cơ.

BÀI TẬP 3: Viết chương trình điều khiển hai động cơ kéo 2 băng tải có cảm biến đếm sản phẩm theo yêu cầu sau:

- Khi nhấn “ON” lần đầu thì cả hai băng tải đều chạy, nhấn “ON” lần thứ hai để dừng khẩn cấp.
- Khi chưa có sản phẩm nào đi qua thì đèn 1 sáng.
- Khi tổng sản phẩm trên 2 băng tải là từ 1 ÷ 10 thì đèn 1 tắt, đèn 2 sáng.
- Khi tổng sản phẩm trên 2 băng tải là từ 11 ÷ 20 thì đèn 2 tắt, đèn 3 sáng.
- Khi tổng sản phẩm trên 2 băng tải là từ 21 ÷ 30 thì đèn 3 tắt, đèn 4 sáng.
- Khi đủ 30 sản phẩm thì hệ thống dừng.
- Khi số sản phẩm qua hai băng tải lệch nhau quá 2 sản phẩm thì băng tải nào có sản phẩm nhiều hơn phải tạm dừng để chờ cho đến khi băng tải kia có cùng số lượng sản phẩm thì mới chạy lại.

BÀI TẬP 4: Viết chương trình điều khiển hai động cơ kéo hai băng tải có hai cảm biến đếm sản phẩm bên trên theo yêu cầu sau:

- Khi nhấn ON thì băng tải 1 chạy, khi băng tải 1 có từ 2 sản phẩm trở lên đi qua thì băng tải thứ 2 chạy.

- Khi nhấn OFF, nếu số sản phẩm đi qua băng tải 1 nhiều hơn số sản phẩm qua băng tải 2 là 2 sản phẩm thì hệ thống dừng, ngược lại thì hệ thống tiếp tục chạy cho đến khi thỏa điều kiện.

- Nhấn STOP để dừng khẩn cấp

BÀI TẬP 5: Viết chương trình điều khiển hai động cơ chạy luân phiên theo yêu cầu sau:

- Mỗi động cơ chạy 30s, khởi động \square/Δ , thời gian khởi động là 5s.

- Chu kỳ hoạt động được chọn bằng các nút nhấn “2lần”, “4lần”, “6lần”. Khi nhấn nút nào thì hệ thống hoạt động tương ứng với chu kỳ đó.

- Khi nhấn “OFF” thì hệ thống dừng sau khi chạy hết 30s.

- Khi nhấn “STOP” thì hệ thống dừng khẩn cấp.

Bài 5: KẾT NỐI HỆ THỐNG ĐIỀU KHIỂN PLC

Mục tiêu của bài:

- Trình bày được qui trình kết nối mô hình bằng PLC.
- Download chương trình về mô hình và kết nối phần cứng đúng qui trình, mô hình hoạt động chính xác.
- Thao tác chính xác, an toàn, đúng kỹ thuật.

Nội dung của bài:

I. Giới thiệu phần cứng kit thí nghiệm S7 – 300:

A. PHẦN LÝ THUYẾT:

PLC S7-300 hiện có tại xưởng tự động hóa là loại CPU 312C, module CPU và các module mở rộng đều được kết nối đến các jack cắm nhằm tiện cho việc thực tập của học sinh, cụ thể như sau:

1.Module CPU:

CPU-312C sử dụng cho hệ thống thí nghiệm với thế nuôi 24VDC, có 10 lối vào số /24VDC, và 6 lối ra số /24VDC. 10 ngõ vào số có địa chỉ từ I0.0 đến I1.1, mỗi ngõ vào được mô phỏng trên mô hình bằng 1 LED đỏ. 6 ngõ ra số 24VDC, có địa chỉ từ Q0.0 đến Q0.5, mỗi ngõ ra cũng được mô phỏng trên mô hình bằng 1 LED đỏ.

Như vậy để tác động ngõ vào CPU thì ta cấp 24VDC tại các ngõ trên mô hình từ Q0.0 đến Q1.1 và khi CPU xuất tín hiệu ra thì các ngõ ra tương ứng trên mô hình từ Q0.0 đến Q0.5 sẽ có điện thế 24VDC.

Các ngõ ra CPU chỉ dùng để điều khiển các thiết bị có nguồn cấp 24VDC (các đèn, LED, Motor DC) mà không dùng đóng ngắt trực tiếp các công tắc tơ, do đó để đóng ngắt các công tắc tơ, ta phải dùng đến roler trung gian.

2. Module mở rộng ngõ vào số (Digital Input Module (DI))

SM 321 DI 16 x DC24V/321 – 1BH02-0AA0):

Để mở rộng lối vào điều khiển cho CPU S7-300, người ta dùng các module ngõ vào mở rộng. Các module mở rộng hiện có tại xưởng thực hành có đặc điểm sau:

- 16 lối vào số (digital) độc lập và cách ly, điện thế cho lối vào 24V/10mA, chỉ thị LED trạng thái vào, trên mô hình còn được mô phỏng thêm bằng LED đỏ
- Điện thế nuôi cho khối (qua chân L+ & M): 24V.

3. Module mở rộng ngõ ra số (Digital Output Module (DO))

SM 322 DO 16xRel. AC120V/230V – 1HH01-0AA0):

Để mở rộng ngõ ra điều khiển cho CPU S7-300, người ta dùng các module ngõ ra mở rộng. Các module mở rộng ngõ ra hiện có tại xưởng thực hành có đặc điểm sau:

- 16 lối ra relay độc lập, dòng giới hạn ở tiếp điểm relay: 2A. Chỉ thị LED trạng thái ra.
- Điện thế nuôi cho khối (qua chân L+ & M): 24V.

4. Analog Input Module SM 331:

- 2 lối vào analog độc lập, phân giải 12 bit,
- Điện thế nuôi cho khối (qua chân L+ & M): 24V.
- Chức năng mở rộng lối vào điều khiển cho CPU S7-300.

5. Analog Output Module SM 332:

- 2 lối ra analog độc lập, phân giải 12 bit,
- Điện thế nuôi cho khối (qua chân L+ & M): 24V. - Chức năng mở rộng lối ra điều khiển cho CPU S7-300.

6. Các khối phụ trợ cho thí nghiệm

Các khối phụ trợ cho thí nghiệm gồm các module chứa công tắc, relay, đèn báo, có cấu trúc như sau:

- Khối Contact LSW-16

Chứa 16 nút nhấn (công tắc đơn), phục vụ cho việc tạo các trạng thái lối vào cho PLC.

- Khối Đèn: Chứa các LED mắc nối tiếp với điện trở, chịu điện thế 24V, sử dụng để hiển thị trạng thái ngõ ra.

- Khối cài đặt ngõ vào và hiển thị ngõ ra bằng LED 7 đoạn: mỗi khối có 16 đường địa chỉ vào(ra) nhằm cài đặt dữ liệu trực tiếp trên mô hình hoặc hiển thị ngõ ra.

- Motor kéo băng tải: có 4 ngõ vào dùng để cấp nguồn và đảo chiều động cơ

- Khối nguồn 24V / 5A: cấp tín hiệu vào cho các ngõ vào số đồng thời là nguồn nuôi cho các modules, các động cơ DC.

B. PHẦN THỰC HÀNH:

Bài 1: Hãy xác định tên, chức năng và các ngõ vào/ra của CPU và các module mở rộng của các PLC hiện có tại xưởng.

* Các bước thực hiện:

- Bước 1: Xác định tên các module và ghi lại các tên này, bắt đầu từ CPU.
- Bước 2: Dựa vào tên, xác định chức năng của từng module.
- Bước 3: Xác định các ngõ vào/ra của các module (I0.0, I0.1.../Q0.0, Q0.1...)

Bài 2: Hãy chỉ ra khu vực bố trí các khối phụ trợ trên bộ thí nghiệm PLC. Xác định đặc tính, chức năng của các khối này.

- Bước 1: Xác định khối công tắc (nút nhấn), tìm chân chung và các chân riêng.
- Bước 2: Xác định khối đèn LED, tìm chân chung và các chân riêng.
- Bước 3: Xác định khối cài đặt ngõ vào và hiển thị ngõ ra bằng LED 7 đoạn.
- Bước 4: Xác định khối nguồn cấp cho các motor DC.

HÌNH 6.1: Kít thí nghiệm S7 – 300.

II. Cách kết nối dây cho hệ thống:

A. PHẦN LÝ THUYẾT:

Sau khi viết chương trình và download về PLC, muốn mô hình hoạt động ta phải kết nối phần cứng theo trình tự các bước sau:

1. Vẽ sơ đồ kết nối:

Dựa vào phần lập trình và phần cứng thực tế, ta vẽ sơ đồ kết nối như sau:

- Các ngõ vào được kết nối đến nguồn 24VDC thông qua các SW hoặc các nút nhấn: Điểm chung (C: Common) của các nút nhấn được kết nối đến nguồn 24VDC, phía còn lại của từng nút nhấn được nối đến các ngõ vào.

- Các ngõ ra xuất nguồn 24VDC: Các ngõ ra được đưa đến các LED hiển thị hoặc cấp nguồn điều khiển chiều quay các motor DC. Ngoài ra, đối với các LED, ta cần phải kết nối chân chung đến mass và đối với động cơ DC ta phải cấp thêm nguồn DC.

- Các ngõ ra của module mở rộng: là các rơle có thể đóng ngắt được cho các nguồn xoay chiều, dùng để tác động các công-tác-tơ.

2. Kết nối mô hình theo sơ đồ đã vẽ: Dựa vào sơ đồ đã vẽ bên trên, ta kết nối hệ thống.

3. Kiểm tra hệ thống đã kết nối: sau khi kết nối, ta kiểm tra hệ thống lần cuối, lưu ý các ngõ vào/ra hệ thống phải được kết nối chính xác, các cọc nối phải chắc chắn, đảm bảo an toàn. Sau khi kiểm tra xong, cho hệ thống vận hành và quan sát quá trình vận hành để chỉnh sửa phần lập trình nếu cần thiết.

4. Vận hành hệ thống: Sau khi kiểm tra xong, cho hệ thống vận hành và quan sát quá trình vận hành để chỉnh sửa phần lập trình (nếu cần) nhằm đảm bảo hệ thống hoạt động chính xác, đúng yêu cầu.

B. PHẦN THỰC HÀNH:

III. Bài tập ứng dụng:

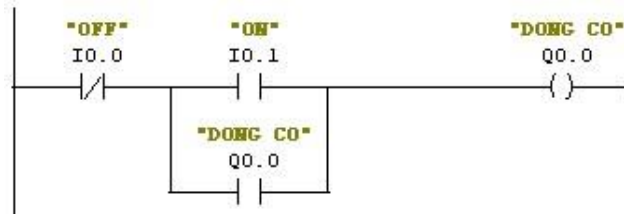
1. Bài tập 1:

Viết chương trình điều khiển mở máy trực tiếp động cơ không đồng bộ 3 pha theo yêu cầu sau:

- Khi nhấn “ON“ thì động cơ hoạt động, nhấn “OFF“ thì động cơ dừng.
- Thiết lập phần cứng cho mô hình.
- Kết nối mô hình cho hệ thống hoạt động.

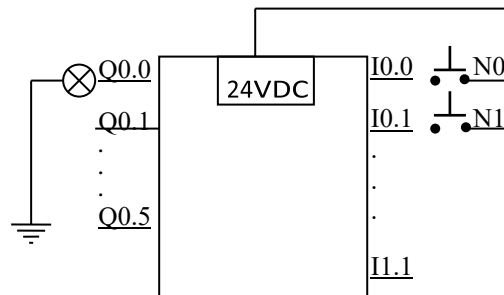
* Các bước thực hiện:

- Thiết lập phần cứng: mở cửa sổ Hardware config để khai báo cấu hình phần cứng theo thiết bị hiện hành tại xưởng.
- Lập trình phần mềm.



Hình 6.2a: Chương trình điều khiển ON/OFF

- Mô phỏng chương trình: mở chương trình mô phỏng, cho chạy thử chương trình.
- Kết nối hệ thống: Vẽ sơ đồ kết nối và kết nối theo sơ đồ



Hình 6.2b: Sơ đồ kết nối phần cứng

Chương trình phần mềm như trên và cách thiết lập phần cứng xem như đã đáp ứng yêu cầu bài tập, việc còn lại là ta kết nối phần cứng sao cho hệ thống dễ vận hành và hoạt động đúng. Theo yêu cầu bài tập thì:

+ Khi cấp 24VDC vào một ngõ vào của PLC (I0.1 chẳng hạn) thì động cơ hoạt động.

+ Khi cấp 24VDC vào một ngõ vào khác của PLC (I0.0 chẳng hạn) khi động cơ đang chạy thì động cơ ngừng hoạt động.

□ Để thỏa hai yêu cầu trên, ta mắc hai nút nhấn thường mở vào sơ đồ kết nối như hình vẽ, khi cần cấp nguồn 24VDC cho ngõ vào nào thì ta nhấn nút được mắc với ngõ vào đó.

+ Khi một ngõ ra tích cực (Q0.0) thì tại đó xuất hiện 24VDC, điện thế này được cấp cho một LED trên mô hình (hoặc bóng đèn 24VDC), cấp mass cho LED thì khi ngõ ra tích cực, LED sáng.

Bước 3: Kiểm tra hệ thống đã kết nối

Bước 4: Vận hành và hiệu chỉnh.

2. Bài tập 2

Viết chương trình điều khiển động cơ chạy thuận 10s, dừng 5s, chạy ngược 10s, dừng 5s. Chu kỳ lặp lại 3 lần thì dừng hẳn.

- Kết nối mô hình cho hệ thống hoạt động.
 - * Các bước thực hiện:
 - Thiết lập phần cứng
 - Lập trình phần mềm.
 - Mô phỏng chương trình
 - Kết nối hệ thống
 - Vận hành hệ thống.

TÀI LIỆU THAM KHẢO

1. Giáo trình PLC S7-200, S7-300 – <http://www.ebook.edu.vn>
2. Giáo trình PLC – Ths. Lê Văn Bạ, KS. Lê Ngọc Bích.
3. Giáo trình S7-200 – Hà Văn Trí – Công ty TNHH TM-DV kỹ thuật SIS.
4. Bài giảng S7-300 – Hà Văn Trí – Công ty TNHH TM-DV kỹ thuật SIS.
5. Bài tập và hướng dẫn giải bài tập lập trình PLC S7-300 – Nguyễn Xuân Công, ĐH SPKT Hưng Yên.